BLOOM: A 176B-Parameter Open-Access Multilingual Language Model

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al.

To cite this version:

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, et al.. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. 2023. hal-03850124

HAL Id: hal-03850124
https://inria.hal.science/hal-03850124
Preprint submitted on 20 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
BLOOM: A 176B-Parameter Open-Access Multilingual Language Model

BigScience Workshop*

Major Contributors

Dataset

Tokenization
Arun Raja, Benjamin Heinerling, Benoît Sagot, Chonglei Si, Colin Raffel, Davut Emre Taşar, Elizabeth Salesky, Lucile Saulnier, Manan Dey, Matthias Gallé, Pedro Ortiz Suarez, Roman Castagné, Sabrina J. Mielke, Samson Tan, Teven Le Scao, Thomas Wang, Wilson Y. Lee, Zaid Alyafeai

Prompt Engineering

* Please direct correspondence to bigscience-contact@googlegroups.com. A list of contributions is available in Section 6.
Xiangru Tang, Zaid Alyafeai, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh

Architecture and Objective

Engineering
Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minja Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Niklas Muennighoff, Noumane Tazi, Olutunji Ruwase, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbandari, Sanchit Gandhi, Shaden Smith, Stas Bekman, Stéphane Requena, Suraj Patil, Teven Le Scao, Thomas Wang, Tim Dettmers

Evaluation and Interpretability

Broader Impacts

Applications
Abhinav Ramesh Kashyap, Albert Villanova del Moral, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Carlos
BLOOM


Organization
Angela Fan, Christopher Akiki, Douwe Kiela, Giada Pistilli, Margot Mieskes, Mathilde Bras, Matthias Gallé, Suzana ilić, Yacine Jernite, Younes Belkada, Thomas Wolf

Abstract
Large language models (LLMs) have been shown to be able to perform new tasks based on a few demonstrations or natural language instructions. While these capabilities have led to widespread adoption, most LLMs are developed by resource-rich organizations and are frequently kept from the public. As a step towards democratizing this powerful technology, we present BLOOM, a 176B-parameter open-access language model designed and built thanks to a collaboration of hundreds of researchers. BLOOM is a decoder-only Transformer language model that was trained on the ROOTS corpus, a dataset comprising hundreds of sources in 46 natural and 13 programming languages (59 in total). We find that BLOOM achieves competitive performance on a wide variety of benchmarks, with stronger results after undergoing multitask prompted finetuning. To facilitate future research and applications using LLMs, we publicly release our models and code under the Responsible AI License.\(^1\)

Keywords: Language models, collaborative research

1. Introduction
Pretrained language models have become a cornerstone of modern natural language processing (NLP) pipelines because they often produce better performance from smaller quantities of labeled data. The development of ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), GPT (Radford et al., 2018), and BERT (Devlin et al., 2019) led to the widespread use of pretrained models as an initialization for finetuning on downstream tasks. The subsequent finding that pretrained language models can perform useful tasks without any additional training (Radford et al., 2019; Brown et al., 2020) further demonstrated their utility. In addition, the empirical observation that a language model’s performance tends to increase as the model is made larger—sometimes predictably (Hestness et al., 2017; Kaplan

\(^1\) hf.co/bigscience/bloom
et al., 2020; Hoffmann et al., 2022) and sometimes suddenly (Wei et al., 2022)—has led to a
trend of increasing scale (Zeng et al., 2021; Rae et al., 2021; Smith et al., 2022; Chowdhery
et al., 2022). Apart from environmental concerns (Strubell et al., 2019; Lacoste et al., 2019;
Schwartz et al., 2020), the costs of training large language models (LLMs) are only afford-
able for well-resourced organizations. Furthermore, until recently, most LLMs were not
publicly released. As a result, the majority of the research community has been excluded
from the development of LLMs. This exclusion has had concrete consequences; for example,
most LLMs are primarily trained on English-language text (with notable exceptions in
Chinese and Korean, e.g. Wang et al., 2021; Zeng et al., 2021; Kim et al., 2021).

To address these issues, we present the BigScience Large Open-science Open-access Mul-
tilingual Language Model (BLOOM, BigScience Workshop, 2022). BLOOM is a 176 billion
parameter language model trained on 46 natural languages and 13 programming languages
that was developed and released by a collaboration of hundreds of researchers. The com-
pute for training BLOOM was provided through a French public grant from GENCI and
IDRIS, leveraging IDRIS’ Jean Zay supercomputer. To build BLOOM, we undertook a
thorough design process for each of its components, including the training dataset (Sec-
tion 3.1), model architecture and training objective (Section 3.2), and engineering strategy
for distributed learning (Section 3.4). We also performed an analysis of the model’s capa-
bilities (Section 4). Our overall aim is not only to publicly release a large-scale multilingual
language model with performance comparable to recently developed systems, but also to
document the coordinated process that went into its development (Section 2.2). The pur-
pose of this paper is to provide a high-level overview of these design steps while referencing
the individual reports we produced over the course of developing BLOOM.

2. Background

Before describing the BLOOM model itself, in this section we provide necessary background
on LLMs as well as an organizational overview of the BigScience effort.

2.1 Language Modeling

Language modeling refers to the task of modeling the probability of a sequence of tokens in a
text (Shannon, 1948), where a token is a unit of text (e.g. word, subword, character or byte,
etc., as discussed by Mielke et al., 2021). In this work (and in most current applications
of language modeling) we model the joint probability of tokens in a text as:

\[ p(x) = p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_{<t}) \]  (1)

where \( x \) is a sequence of tokens, \( x_t \) is the \( t \)th token, and \( x_{<t} \) is the sequence of tokens
preceding \( x_t \). This approach is referred to as autoregressive language modeling and can be
seen as iteratively predicting the probability of the next token.

Early Language Models  Language models have a long history of application in NLP.
Early language models (such as those developed by Shannon, 1948) were primarily \( n \)-gram
models that estimate the probability of a length-\( n \) sequence of tokens in accordance with
the number of times it appears in a training corpus. In practice, $n$-gram models face two major issues: first, they grow exponentially in size as $n$ is increased; and second, they have no direct way of producing a probability for a sequence of tokens that does not appear in their training data. Advances on these problems enabled $n$-gram models to see widespread use across most areas of NLP (Goodman, 2001).

**Neural Language Models** An alternative to $n$-gram models, first proposed by Miikkulainen and Dyer (1991) and Schmidhuber and Heil (1996) and later popularized by Bengio et al. (2000), is to use a neural network to estimate the probability of the next token given prior tokens. While early work used feed-forward networks with a fixed-length history window, Mikolov et al. (2010); Sutskever et al. (2011); Graves (2013) proposed to use recurrent neural networks instead and found that this significantly improved performance. More recently, language models based on the Transformer architecture (Vaswani et al., 2017) were shown to be more effective than recurrent neural networks (Radford et al., 2018; Al-Rfou et al., 2019; Kaplan et al., 2020). Consequently, the Transformer has become the de facto choice for language models.

**Transfer Learning** In tandem with advances in language modeling using neural networks, NLP pipelines have increasingly adopted the framework of transfer learning. In transfer learning, the parameters of a model are first pretrained on a data-rich task before being finetuned on a downstream task. A historically common approach to obtaining pretrained parameters were word vectors (Mikolov et al., 2013) trained so that the dot product of co-occurring word vectors is large. However, subsequent work by Peters et al. (2018); Howard and Ruder (2018); Radford et al. (2018); Devlin et al. (2019) showed that the framework of Collobert et al. (2011), where the entire model is pretrained before being finetuned, can attain stronger performance. In particular, Radford et al. (2018); Devlin et al. (2019) demonstrated strong results using pretrained Transformer language models, prompting work on progressively better models (Liu et al., 2019; Yang et al., 2019; Lewis et al., 2020; Raffel et al., 2020; Zhang et al., 2019, etc.).

**Few- and Zero-Shot Learning** While finetuning a pretrained model remains an effective way of attaining high performance with limited labeled data, a parallel line of work has demonstrated that pretrained language models can be induced to perform tasks without any subsequent training. After Vinyals and Le (2015) observed limited task-performing behavior in a neural dialog model, Radford et al. (2019) later demonstrated that Transformer-based language models trained on text scraped from the web could perform various tasks to varying degrees. Notably, Radford et al. (2019) found that performance improved with model scale, inspiring work to characterize (Kaplan et al., 2020; Hoffmann et al., 2022) and exploit (Shoeybi et al., 2019; Brown et al., 2020; Smith et al., 2022; Chowdhery et al., 2022; Rae et al., 2021; Wang et al., 2021; Zeng et al., 2021; Zhang et al., 2022) the benefits of scale. A major factor in the success of this approach is the way that task-specific examples are formatted when fed into the model. Brown et al. (2020) popularized the idea of designing “prompts” that provide natural-language descriptions of the task and also allow inputting a few demonstrations of input-output behavior.

**Social Limitations of LLM Development** While the continued increase in the size of large language models has resulted in improvements across a wide range of tasks, it has also
The computational expense of large models also prohibits the majority of the research community from participating in their development, evaluation and routine use. Moreover, the computational costs have also lead to concerns about the carbon footprint stemming from the training and use of large language models (Strubell et al., 2019; Lacoste et al., 2019; Schwartz et al., 2020; Bannour et al., 2021), and existing carbon footprint studies have likely under-estimated emissions (Bannour et al., 2021). Contributing to an increase in the global carbon footprint exacerbates climate change which most severely affects already-marginalized communities (Westra and Lawson, 2001). Furthermore, the concentration of resources within a handful of (typically industrial) institutions with primarily technical expertise hinders prospects for an inclusive, collaborative, and reliable governance of the technology. First, public narratives about the technology that are driven by industry actors can lead to inflated expectations about its suitability for use (Brennen, 2018; Brennen et al., 2022), leading to misaligned research and policy priorities (Raji et al., 2022) and potentially dire consequences in e.g. medical applications (Wong et al., 2021). Second, in a world mediated by technology, choices at all stages of its development end up shaping people’s lives in a way that can be most closely compared to regulations (Winner, 1977, 2017), albeit without the same explicit consultation of stakeholders in the process. When the development efforts are guided by prioritizing internal definitions of performance over their impact on society, the values of the developers come to be emphasized over those of the direct and indirect users (Birhane et al., 2022). Despite the substantial social dangers in allowing this technology to be developed unilaterally by corporations, EleutherAI (Phang et al., 2022) was the only non-corporate entity outside of China that was developing large language models before the BigScience Workshop was convened.

2.2 BigScience

Participants BLOOM’s development was coordinated by BigScience, an open research collaboration whose goal was the public release of an LLM. The project started after being awarded by GENCI a compute grant on its Jean Zay supercomputer at IDRIS/CNRS. It was initially built around a concerted effort from Hugging Face and the French NLP community (the “founding members”), and quickly opened up to grow into a broader international collaboration to support its aims of linguistic, geographical, and scientific diversity. In the end, over 1200 people registered as participants in BigScience and were given access to its communication channels. They had background not only in machine learning and computer science, but also linguistics, statistics, socio-cultural anthropology, philosophy, law, and other fields. Of those, hundreds of individuals have directly contributed to one of the project’s released artifacts. While the largest number of participants ultimately originated from the US, 38 countries were represented.

Organization The set of related research questions tackled by the BigScience effort was reflected in the project’s organization into working groups. Each working group comprised several participants with various levels of involvement, including chairs whose role was to self-organize around a specific aspect of the overall project. Importantly, participants were encouraged to join more than one working group in order to share experiences and information, which resulted in the set of 30 working groups presented in Figure 1. Most
of the working groups focused on tasks directly linked to the development of BLOOM. In addition, a few groups focused on the evaluation of LLMs and dataset development in specific domains, such as biomedical texts (Fries et al., 2022b) and historical texts (De Toni et al., 2022). A larger overview of the motivations behind this initiative, its history and some of the lessons learned can be found in Akiki et al. (2022).

![Figure 1: Organization of BigScience working groups.](image)

**Ethical Considerations within BigScience** In order to acknowledge and start addressing social limitations of LLM development within BigScience, the workshop relied on a collaboratively designed Ethical Charter and original research on applicable regulations in jurisdictions outside of the US to guide its choices throughout the project. In particular, the charter emphasizes values of inclusivity and diversity, openness and reproducibility, and responsibility in various aspects of the organization (Akiki et al., 2022). Each of these values are showcased in different ways in the dataset curation (Section 3.1), modeling (Section 3.2), engineering (Section 3.4), evaluation (Section 4), and other social impact (throughout) aspects of the project.

3. BLOOM

In this section, we document the design of BLOOM, including its training dataset (Section 3.1), architecture (Section 3.2), tokenizer (Section 3.3), computing infrastructure (Section 3.4), and training hyperparameters (Section 3.5).

3.1 Training Dataset

BLOOM was trained on the ROOTS corpus (Laurençon et al., 2022), a composite collection of 498 Hugging Face datasets (Lhoest et al., 2021) amounting to 1.61 terabytes of text that span 46 natural languages and 13 programming languages. A high-level overview of this dataset can be seen in Figure 3, while a detailed itemized list of every language along with its linguistic genus, family and macroarea is presented in Table 1. Beyond the corpus itself, the process resulted in the development and release of a number of organizational and technical tools, including those illustrated in Figure 2. The rest of this section will

---

2. bigscience.huggingface.co/blog/bigscience-ethical-charter
3. bigscience.huggingface.co/blog/legal-playbook-for-natural-language-processing-researchers
<table>
<thead>
<tr>
<th>Language</th>
<th>ISO-639-3</th>
<th>catalog-ref</th>
<th>Genus</th>
<th>Family</th>
<th>Macroarea</th>
<th>Size in Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akan</td>
<td>aka</td>
<td>ak</td>
<td>Kwa</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>70,155</td>
</tr>
<tr>
<td>Arabic</td>
<td>arb</td>
<td>ar</td>
<td>Semitic</td>
<td>Afro-Asiatic</td>
<td>Eurasia</td>
<td>74,854,900,600</td>
</tr>
<tr>
<td>Assamese</td>
<td>asm</td>
<td>as</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>291,522,098</td>
</tr>
<tr>
<td>Bambara</td>
<td>bnm</td>
<td>bnm</td>
<td>Western Mande</td>
<td>Mande</td>
<td>Africa</td>
<td>391,747</td>
</tr>
<tr>
<td>Basque</td>
<td>eus</td>
<td>eu</td>
<td>Basque</td>
<td>Basque</td>
<td>Eurasia</td>
<td>2,360,470,848</td>
</tr>
<tr>
<td>Bengali</td>
<td>ben</td>
<td>bn</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>18,606,823,104</td>
</tr>
<tr>
<td>Catalán</td>
<td>cat</td>
<td>ca</td>
<td>Romance</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>17,792,493,289</td>
</tr>
<tr>
<td>Chichewa</td>
<td>nya</td>
<td>ny</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>1,187,405</td>
</tr>
<tr>
<td>chiShona</td>
<td>sna</td>
<td>sn</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>6,638,639</td>
</tr>
<tr>
<td>Chitumbuka</td>
<td>tum</td>
<td>tun</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>170,360</td>
</tr>
<tr>
<td>English</td>
<td>eng</td>
<td>en</td>
<td>Germanic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>484,953,009,124</td>
</tr>
<tr>
<td>Fon</td>
<td>fon</td>
<td>fon</td>
<td>Kwa</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>2,478,546</td>
</tr>
<tr>
<td>French</td>
<td>fra</td>
<td>fr</td>
<td>Romance</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>208,242,620,434</td>
</tr>
<tr>
<td>Gujarati</td>
<td>guj</td>
<td>gu</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>1,199,986,460</td>
</tr>
<tr>
<td>Hindi</td>
<td>hin</td>
<td>hi</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>24,622,119,985</td>
</tr>
<tr>
<td>Igbo</td>
<td>ibo</td>
<td>ig</td>
<td>Igboid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>14078,521</td>
</tr>
<tr>
<td>Indonesian</td>
<td>ind</td>
<td>id</td>
<td>Malayo-Sumbawan</td>
<td>Austronesian</td>
<td>Papunesia</td>
<td>19,972,325,222</td>
</tr>
<tr>
<td>isiXhosa</td>
<td>xho</td>
<td>xh</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>14,304,074</td>
</tr>
<tr>
<td>isiZulu</td>
<td>zul</td>
<td>zu</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>8,511,561</td>
</tr>
<tr>
<td>Kannada</td>
<td>kan</td>
<td>kn</td>
<td>Southern Dravidian</td>
<td>Dravidian</td>
<td>Eurasia</td>
<td>2,998,453,560</td>
</tr>
<tr>
<td>Kikuyu</td>
<td>kik</td>
<td>ki</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>359,615</td>
</tr>
<tr>
<td>Kinyarwanda</td>
<td>kin</td>
<td>rw</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>40,428,299</td>
</tr>
<tr>
<td>Kirundi</td>
<td>run</td>
<td>rn</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>3,272,550</td>
</tr>
<tr>
<td>Lingala</td>
<td>lin</td>
<td>ln</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>1,650,804</td>
</tr>
<tr>
<td>Luganda</td>
<td>lug</td>
<td>lg</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>4,568,367</td>
</tr>
<tr>
<td>Malayalam</td>
<td>mal</td>
<td>ml</td>
<td>Southern Dravidian</td>
<td>Dravidian</td>
<td>Eurasia</td>
<td>3,662,571,498</td>
</tr>
<tr>
<td>Marathi</td>
<td>mar</td>
<td>mr</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>1,775,483,122</td>
</tr>
<tr>
<td>Nepali</td>
<td>nep</td>
<td>ne</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>2,551,307,393</td>
</tr>
<tr>
<td>Northern Sotho</td>
<td>nso</td>
<td>nso</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>1,764,506</td>
</tr>
<tr>
<td>Odia</td>
<td>ori</td>
<td>or</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>1,157,100,133</td>
</tr>
<tr>
<td>Portuguese</td>
<td>por</td>
<td>pt</td>
<td>Romance</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>79,277,543,375</td>
</tr>
<tr>
<td>Punjabi</td>
<td>psn</td>
<td>ps</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>1,572,109,752</td>
</tr>
<tr>
<td>Sesotho</td>
<td>sot</td>
<td>st</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>751,034</td>
</tr>
<tr>
<td>Setswana</td>
<td>tsn</td>
<td>tn</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>1,502,200</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>zhs</td>
<td>Chinese</td>
<td>Sino-Tibetan</td>
<td>Eurasia</td>
<td>261,019,433,892</td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td>spa</td>
<td>es</td>
<td>Romance</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>175,998,365,045</td>
</tr>
<tr>
<td>Swahili</td>
<td>swh</td>
<td>sw</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>236,482,543</td>
</tr>
<tr>
<td>Tamil</td>
<td>tam</td>
<td>ta</td>
<td>Southern Dravidian</td>
<td>Dravidian</td>
<td>Eurasia</td>
<td>7,989,206,220</td>
</tr>
<tr>
<td>Telugu</td>
<td>tel</td>
<td>te</td>
<td>South-Central Dravidian</td>
<td>Dravidian</td>
<td>Eurasia</td>
<td>2993407,159</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>zht</td>
<td>Chinese</td>
<td>Sino-Tibetan</td>
<td>Eurasia</td>
<td>762,489,150</td>
<td></td>
</tr>
<tr>
<td>Twi</td>
<td>twi</td>
<td>tw</td>
<td>Kwa</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>1,265,041</td>
</tr>
<tr>
<td>Urdu</td>
<td>urd</td>
<td>ur</td>
<td>Indic</td>
<td>Indo-European</td>
<td>Eurasia</td>
<td>2,781,329,959</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>vie</td>
<td>vi</td>
<td>Viet-Muong</td>
<td>Austro-Asiatic</td>
<td>Eurasia</td>
<td>43,709,279,959</td>
</tr>
<tr>
<td>Wolof</td>
<td>wol</td>
<td>wo</td>
<td>Wolof</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>3,606,973</td>
</tr>
<tr>
<td>Xitsongta</td>
<td>tso</td>
<td>ts</td>
<td>Bantoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>707,634</td>
</tr>
<tr>
<td>Yoruba</td>
<td>yor</td>
<td>yo</td>
<td>Defoid</td>
<td>Niger-Congo</td>
<td>Africa</td>
<td>89,695,835</td>
</tr>
</tbody>
</table>

| Programming Languages | —       | —          | —          | —                  | —               | 174,700,245,772 |

Table 1: Linguistic makeup of the ROOTS corpus.
contextualize these efforts by providing a brief summary of the steps taken to compile the corpus. For more detailed documentation of the overall dataset curation process and its outcomes, we refer the reader to Laurençon et al. (2022).

**Motivation**  The disconnect between developers and (in)voluntary users of the technology mentioned in Section 2 is particularly apparent in the curation of the datasets that have supported recent large-scale machine learning projects, where intentional “Data work” is generally under-valued (Sambasivan et al., 2021). In the context of LLMs, this tendency is exemplified by a range of heuristics-based filtering approaches that prioritize getting as much “high-quality” data for as little cost as possible over engaging with the needs—and rights—of data subjects, where quality is commonly defined as maximizing performance on downstream tasks while occasionally removing content deemed offensive by the developers.

While these approaches do yield terabytes of data with comparatively little human effort, compounding biases of the source material (such as CommonCrawl dumps) with those of the filtering method often leads to negative outcomes for marginalized populations. In one case, the use of a block list to remove “pornographic” text was shown to also suppress LGBTQ+ and African American English (AAE) text from a corpus (Dodge et al., 2021). In another, using Reddit outgoing links as an indicator of quality for a seed corpus (Radford et al., 2019) leads to trained models that implicitly prioritize US-centric views in their outputs (Johnson et al., 2022). In yet another project, a filtering approach that relied on a machine learning image-text alignment model was shown to exacerbate its biases in the created multimodal dataset (Birhane et al., 2021). In addition, this abstractive approach to data curation leads to corpora that are difficult to meaningfully document and govern after the fact, as the provenance and authorship of individual items is usually lost in the process (although works such as Gao et al. (2020) that prioritize compilations of previously documented individual sources over crawled data provide a step towards addressing these issues (Biderman et al., 2022)).

In the context of the BigScience workshop, and in accordance with its Ethical Charter, we aimed to prioritize human involvement, local expertise, and language expertise in our data curation and documentation process, as outlined in the following sections.

### 3.1.1 Data Governance

Large text corpora comprise text about and created by people: the data subjects. Different people and institutions might legally “own” that data, making them data rights-holders. As machine learning developers gather and collate that data into ever-larger datasets to support training larger models, it becomes increasingly important to develop new ways of accounting for the interests of all parties involved – developers, data subjects, and rights-holders alike.

The BigScience effort aimed to address these needs through a multidisciplinary lens combining technical, legal, and sociological expertise. The group focused on two main interrelated goals at two different time scales: the design of a structure for long-term international data governance that prioritizes the agency of the data rights-holders, and concrete recommendations for handling the data used directly in the BigScience project. Progress on the first goal is presented in the work of Jernite et al. (2022), which further motivates the needs and requirements of data governance, and outlines the structure needed for a network

---

4. [bigscience.huggingface.co/blog/bigscience-ethical-charter](bigscience.huggingface.co/blog/bigscience-ethical-charter)
of data custodians, rights-holders, and other parties to appropriately govern shared data. The interactions between these actors are designed to account for the privacy, intellectual property, and user rights of the data and algorithm subjects in a way that aims to prioritize local knowledge and expression of guiding values. In particular, this approach relies on structured agreements between data providers and data hosts\(^5\) that specify what the data may be used for.

While we were not able to fully establish an international organization in the comparatively short time between the project start and model training, we worked on integrating lessons from this effort (and conversely adapting it to the practical concerns we were experiencing) in the following main ways: (i) we sought explicit permission to use the data from specific providers within the context of BigScience whenever possible (such as for the AI2\(^6\)-managed S2ORC corpus of Lo et al. (2020) or articles from the French newspaper Le Monde\(^7\)); (ii) we kept individual sources separate until the final stages of preprocessing to maintain traceability and handle each according to the needs of its specific context; and (iii) we adopted a composite release approach for the various data sources that make up the overall corpus to foster reproducibility and follow-up research while respecting these source-dependent needs. Resources to visualize and access the ROOTS corpus can be found on the Hugging Face Hub organization “BigScience Data”\(^8\). The organization hosts several demos (or “Spaces”) that can be used to gain insights into the full corpus, as well as direct access to the 223 (out of 498) components that we are able to distribute taking into account their licensing status, privacy risks, and agreements with their original custodians. Finally, since we understand that future investigation into the BLOOM models may require full access to the entire corpus, we are also inviting researchers with a relevant research project in mind to join ongoing efforts to analyze the data through a sign-up form\(^9\).

3.1.2 Data Sources

Given a strategy for data governance, the next step was to determine the composition of the training corpus. This stage was driven by several goals, which sometimes had inherent tensions. Some of those tensions included building a language model that was accessible to as many people as possible around the world while only including languages for which we had enough expertise to curate a dataset of comparable scale (and to a lesser extent composition) to previous efforts while improving the standards of documentation and respect for data and algorithm subject rights.

Language Choices These considerations led us to an incremental process for choosing which languages were to be included in the corpus. We started with a list of eight of the world’s largest languages by number of speakers for which we did active outreach in the early stages of the project to invite fluent speakers to join the data efforts. Then, on the recommendation of language communities (Nekoto et al., 2020) we expanded Swahili in the original selection to the category of Niger-Congo languages, and Hindi and Urdu to

5. hf.co/spaces/bigscience/data_host_provider_agreement  
6. allenai.org  
7. lemonde.fr  
8. hf.co/bigscience-data  
9. forms.gle/qyYswbEL5kA23Wu99
Indic languages (Kunchukuttan et al., 2020). Finally, we proposed that any group of 3 or more participants fluent in an additional language could add it to the supported list if they would commit to selecting sources and guiding processing choices in the language in order to avoid common issues with corpora selected through automatic language identification without specific language expertise (Caswell et al., 2022).

**Source Selection** The biggest part of the corpus was curated by workshop participants and research collectives who collectively compiled the “BigScience Catalogue”: a large list of processed and non-processed sources covering a wide range of languages. This took the form of hackathons that were co-organized by communities such as Machine Learning Tokyo, Masakhane, and LatinX in AI (McMillan-Major et al., 2022). Complementary to those efforts, other working group participants compiled language-specific resources such as the Arabic-focused Masader repository (Alyafeai et al., 2021; Altaher et al., 2022). A total of 252 sources were identified through this bottom-up approach, with at least 21 sources per language category. Additionally, in order to increase the geographic coverage of some of our Spanish, Chinese, French, and English sources, participants identified locally relevant websites in their language to be added to the corpus via pseudocrawl, a method to obtain those websites from a Common Crawl snapshot.

**GitHub Code** The catalogue was further complemented with a dataset of programming languages collected from the GitHub data collection on Google’s BigQuery, which was then deduplicated of exact matches. The choice of languages to include mirrored the design choices introduced by Li et al. (2022) to train the AlphaCode model.

**OSCAR** Both in an effort not to diverge from the standard research practice of using the Web as a source of pretraining data (Radford et al., 2018; Raffel et al., 2020), and also to satisfy the data volume needs of our compute budget given the size of BLOOM, we further sourced data from OSCAR version 21.09, corresponding to the February 2021 snapshot of the Common Crawl (Ortiz Suárez et al., 2019; Abadji et al., 2021), which ended up constituting 38% of the corpus.

### 3.1.3 DATA PREPROCESSING

After the sources had been identified, data processing involved several steps to handle multiple aspects of data curation. An overarching view of and processing pipeline to build ROOTS can be seen in Figure 2. All tools developed in the process are available on GitHub.11

**Obtaining the Source Data** The first step involved obtaining the data for all of the text data sources identified in Section 3.1.2, which consisted of a combination of downloading and extracting the text field from a variety of NLP datasets in various formats (including e.g. question answering, summarization, or dialogue datasets), scraping and processing large amounts of PDF files from archives (e.g. the French repository of scientific articles), and extracting and preprocessing text from 192 website entries from the catalogue and another.

---

10. cloud.google.com/blog/topics/public-datasets/github-on-bigquery-analyze-all-the-open-source-code
11. github.com/bigscience-workshop/data-preparation
12. hal.archives-ouvertes.fr
geographically diverse set of 456 websites selected by data working group members. The latter required the development of new tools to extract text from the HTML in the Common Crawl WARC files, which we made available on the main data preparation repository.\textsuperscript{13} We were able to find and extract usable text data from all URLs present in 539 of the websites.

\textbf{“Quality” filtering: Text Produced by Humans for Humans} \hspace{1em} After obtaining the text, we found that most of the sources contained some amount of text that was not natural language, for example preprocessing errors, SEO pages, or spam (including pornographic spam). In order to filter non-natural language, we defined a set of quality indicators, where high-quality text is defined as “written by humans for humans”, without distinction of content (as we wanted content selection to exclusively be the domain of the more accountable human source selection) or \textit{a priori} judgments of grammaticality. The full list of indicators are described in (Laurençon et al., 2022). Importantly, the indicators were adapted to the needs of each of the sources in two main ways. First, their parameters such as the thresholds and supporting term lists were selected individually for each language by fluent speakers. Second, we manually went through each individual source to identify which indicators were most likely to identify non-natural language. Both processes were supported by tools to visualize their impact.\textsuperscript{14,15}

\textsuperscript{13} github.com/bigscience-workshop/data-preparation/tree/main/sourcing/cc_pseudo_crawl
\textsuperscript{14} hf.co/spaces/huggingface/text-data-filtering
\textsuperscript{15} hf.co/spaces/bigscience-data/process-pipeline-visualizer
**Deduplication and Privacy Redaction**  Finally, we removed near-duplicate documents with two deduplication steps and redacted Personal Identifiable Information (such as social security numbers) that we could identify from the OSCAR version of the corpus—as it was deemed to be the source that presented the highest privacy risks, prompting us to apply regex-based redaction even in cases where the expressions had some false positives.

3.1.4 Prompted Datasets

Multitask prompted finetuning (also referred to as instruction tuning) involves finetuning a pretrained language model on a training mixture composed of a large set of different tasks specified through natural language prompts. T0 (Sanh et al., 2022) (developed as part of BigScience) demonstrated that language models finetuned on a multitask mixture of prompted datasets have strong zero-shot task generalization abilities. Moreover, T0 was shown to outperform language models that are an order of magnitude larger but did not undergo such finetuning. Motivated by these results, we explored using existing natural language datasets to carry out multitask prompted finetuning.
T0 was trained on a subset of the Public Pool of Prompts (P3), a collection of prompts for various existing and open-source English natural language datasets. This collection of prompts was created through a series of hackathons involving BigScience collaborators and where hackathon participants wrote a total of of 2000+ prompts for 170+ datasets. Datasets in P3 cover a variety of natural language task including sentiment analysis, question answering, and natural language inference and exclude harmful content or non-natural language such as programming languages. PromptSource (Bach et al., 2022), an open-source toolkit (also developed as part of BigScience) facilitated creating, sharing and using natural language prompts. Full details of the collection process are given in (Sanh et al., 2022; Bach et al., 2022).

After pretraining BLOOM, we applied the same massively multitask finetuning recipe to equip BLOOM with multilingual zero-shot task generalization abilities. We refer to the resulting models as BLOOMZ. To train BLOOMZ, we extended P3 to include new datasets in languages other than English and new tasks, such as translation. This resulted in xP3, a collection of prompts for 83 datasets covering 46 languages and 16 tasks. As highlighted in Figure 4, xP3 mirrors the language distribution of ROOTS. Tasks in xP3 are both cross-lingual (e.g. translation) and monolingual (e.g. summarization, question answering). We used PromptSource to collect these prompts, adding additional metadata to the prompts, such as input and target languages. To study the importance of multilingual prompts, we also machine-translated English prompts in xP3 to the respective dataset languages to produce a collection called xP3mt. Further details on the prompt collection for xP3 and xP3mt are given in Muennighoff et al. (2022b).

3.2 Model Architecture

This section discusses our design methodology and the architecture of the BLOOM model. In-depth studies and experiments can be found in Le Scao et al. (2022) and Wang et al. (2022a). We first review our design methodology, then motivate our choice of training a causal decoder-only model. Finally, we justify the ways that our model architecture deviates from standard practice.

3.2.1 Design Methodology

The design space of possible architectures is immense, making exhaustive exploration impossible. One option would be to exactly replicate the architecture of an existing large language model. On the other hand, a great deal of work on improving existing architectures has seen relatively little adoption (Narang et al., 2021); adopting some of these recommended practices could yield a significantly better model. We take a middle ground and focus on model families that have been shown to scale well, and that have reasonable support in publicly available tools and codebases. We ablate components and hyperparameters of the models, seeking to make the best use of our final compute budget.

Experimental Design for Ablations One of the main draws of LLMs has been their ability to perform tasks in a “zero/few-shot” way: large enough models can perform novel tasks simply from in-context instructions and examples (Radford et al., 2019), without ded-
icated training on supervised samples. Accordingly, and because finetuning a 100B+ model is unwieldy, we focused our evaluation of architectural decisions on zero-shot generalization, and do not consider transfer learning. Specifically, we measured zero-shot performance on diverse aggregates of tasks: 29 tasks from the EleutherAI Language Model Evaluation Harness (EAI-Eval, Gao et al. (2021)), and 9 tasks from the evaluation set of T0 (T0-Eval, Sanh et al. (2022)). There is significant overlap between the two: only one task from T0-Eval (StoryCloze) is not in EAI-Eval, although all prompts between the two are different. See Le Scao et al. (2022) for a detailed list of tasks and baselines. We also note that our tasks aggregates share 17 of the 31 tasks of the evaluation of GPT-3 (Brown et al., 2020).

We conducted our ablation experiments using smaller models. We used the 6.7B parameter scale for the pretraining objective ablations (Wang et al., 2022a) and the 1.3B scale for the rest including position embeddings, activations, and layer normalization (Le Scao et al., 2022). Recently, Dettmers et al. (2022) identified a phase transition for models larger than 6.7B, in which the emergence of “outliers features” is observed. This questions whether results obtained at the 1.3B scale should be assumed to extrapolate to our final model size.

Out-of-scope Architectures We did not consider mixture-of-experts (MoE) (Shazeer et al., 2017), due to a lack of widely used GPU-based codebases suitable for training them at scale. Similarly, we also did not consider state-space models (Gu et al., 2020). At the time of the design of BLOOM, they consistently underperformed in natural language tasks (Gu et al., 2021). Both of these approaches are promising, and have now demonstrated competitive results—at large scales for MoE (Fedus et al., 2022; Srivastava et al., 2022), and at smaller scale for state-space models with H3 (Fu et al., 2023).

3.2.2 Architecture and Pretraining Objective

Although most modern language models are based on the Transformer architecture, there are significant deviations between architectural implementations. Notably, while the original Transformer is based on an encoder-decoder architecture, many popular models have opted for encoder-only (e.g. BERT, (Devlin et al., 2019)) or decoder-only (e.g. GPT, (Radford et al., 2018)) approaches. Currently, all state-of-the-art language models over 100 billion parameters are causal decoder-only models (Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2022). This is in opposition to the findings of Raffel et al. (2020), in which encoder-decoder models significantly outperform decoder-only models for transfer learning.

Prior to our work, the literature was lacking a systematic evaluation of the zero-shot generalization capabilities of different architectures and pretraining objectives. We explored this question in Wang et al. (2022a) where we evaluated encoder-decoder and decoder-only architectures and their interactions with causal, prefix, and masked language modeling pretraining objectives. Our results show that immediately after pretraining, causal decoder-only models performed best – validating the choice of state-of-the-art LLMs. Furthermore, they can be more efficiently adapted after pretraining to a non-causal architecture and objective—an approach which has been further explored and confirmed by Tay et al. (2022).

3.2.3 Modeling Details

Beyond choosing an architecture and pretraining objective, a number of changes to the original Transformer architecture have been proposed. For example, alternative positional
embedding schemes (Su et al., 2021; Press et al., 2021) or novel activation functions (Shazeer, 2020). We thus performed a series of experiments to evaluate the benefit of each of these modifications for a causal decoder-only model in Le Scao et al. (2022). We adopted two architectural deviations in BLOOM:

**ALiBi Positional Embeddings** Instead of adding positional information to the embedding layer, ALiBi directly attenuates the attention scores based on how far away the keys and queries are (Press et al., 2021). Although ALiBi was initially motivated by its ability to extrapolate to longer sequences, we found it also led to smoother training and better downstream performance even at the original sequence length – outperforming both learned (Vaswani et al., 2017) and rotary (Su et al., 2021) embeddings.

**Embedding LayerNorm** In preliminary experiments training a 104B parameters model, we experimented with an additional layer normalization immediately after the embedding layer – as recommended by the bitsandbytes\(^\text{17}\) library (Dettmers et al., 2022) with its StableEmbedding layer. We found this significantly improved training stability. Even though we also found it penalizes zero-shot generalization in Le Scao et al. (2022), we train BLOOM with an additional layer normalization after the first embedding layer to avoid training instabilities. Note the preliminary 104B experiments were conducted in float\(^16\), while the final training was in bfloat\(^16\). Since then, float\(^16\) has been attributed as being responsible for many of the observed instabilities in training LLMs (Zhang et al., 2022; Zeng et al., 2022). It is possible that bfloat\(^16\) alleviates the need for the embedding LayerNorm.

We represent the full architecture of BLOOM in figure 5 for reference.

Figure 5: The BLOOM architecture. The $k_{\text{head}}$ slope parameters for ALIBI are taken as $2^{-\frac{8i}{n}}$ with $n$ the number of heads and $i \in 1, 2, \ldots, n$.

\(^{17}\)github.com/TimDettmers/bitsandbytes

---

16
3.3 Tokenization

The design decisions when training a tokenizer are often neglected in favour of “default” settings (Mielke et al., 2021). For instance, OPT (Zhang et al., 2022) and GPT-3 (Brown et al., 2020) both use GPT-2’s tokenizer, trained for English. This can be justified by the fact that evaluating the impact of a particular choice on the downstream performance of the model is constrained by the large computational costs of training. However, the diverse nature of BLOOM’s training data requires careful design choices to ensure that the tokenizer encodes sentences in a lossless manner.

Validation  We use the fertility (Ács, 2019) of our tokenizer compared to existing monolingual tokenizers as a metric for sanity checks. Fertility is defined as the number of subwords created per word or per dataset by the tokenizer, which we measured using subsets of Universal Dependencies 2.9 (Nivre et al., 2017) and OSCAR (Ortiz Suárez et al., 2019) in the languages of interest. A very high fertility on a language compared to a monolingual tokenizer may indicate a degradation on the downstream multilingual performance of the model (Rust et al., 2021). Our goal was to not degrade the fertility on each language by more than 10 percentage points when comparing our multilingual tokenizer with monolingual tokenizers in corresponding languages. For all experiments, the Hugging Face Tokenizers library (Moi et al., 2019) was used to design and train the tested tokenizers.

<table>
<thead>
<tr>
<th>Tokenizer</th>
<th>fr</th>
<th>en</th>
<th>es</th>
<th>zh</th>
<th>hi</th>
<th>ar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolingual</td>
<td>1.30</td>
<td>1.15</td>
<td>1.12</td>
<td>1.50</td>
<td>1.07</td>
<td>1.16</td>
</tr>
<tr>
<td>BLOOM</td>
<td>1.17 (-11%)</td>
<td>1.15 (+0%)</td>
<td>1.16 (+3%)</td>
<td>1.58 (+5%)</td>
<td>1.18 (+9%)</td>
<td>1.34 (+13%)</td>
</tr>
</tbody>
</table>

Table 2: Fertilities obtained on Universal Dependencies treebanks on languages with existing monolingual tokenizers. The monolingual tokenizers we used were the ones from CamemBERT (Martin et al., 2020), GPT-2 (Radford et al., 2019), DeepESP/gpt2-spanish, bert-base-chinese, monsoon-nlp/hindi-bert and Arabic BERT (Safaya et al., 2020), all available on the HuggingFace Hub.

Tokenizer Training Data  We initially used a non-deduplicated subset of ROOTS. However, a qualitative study on the vocabulary of the tokenizer revealed issues in its training data. For instance, in earlier versions of the tokenizer, we found entire URLs stored as tokens caused by several documents containing a high number of duplicates. These issues motivated us to remove duplicated lines in the tokenizer training data. We then applied the same sampling ratios per language as for the training data.

Vocabulary Size  A large vocabulary size reduces the risk of over-segmenting some sentences, especially for low-resource languages. We conducted validation experiments using 150k and 250k vocabulary sizes to make comparisons with existing multilingual modeling literature easier (Conneau et al., 2020; Xue et al., 2021). We ultimately settled for a vocabulary of 250k tokens to reach our initial fertility objective compared to monolingual tokenizers. Since the vocabulary size determines the embedding matrix size, it also had to
be divisible by 128 for GPU efficiency reasons and by 4 to be able to use Tensor Parallelism. We used a final size of 250,680 vocabulary items with 200 tokens reserved for possible future applications such as removing private information using placeholder tokens.

**Byte-level BPE** The tokenizer is a learned subword tokenizer trained using the Byte Pair Encoding (BPE) algorithm introduced by Gage (1994). In order not to lose information during tokenization, the tokenizer creates merges starting from bytes as the smallest units instead of characters (Radford et al., 2019). This way, tokenization never results in unknown tokens because all 256 bytes can be contained in the vocabulary of the tokenizer. In addition, Byte-level BPE maximizes vocabulary sharing between languages (Wang et al., 2020).

**Normalization** Upstream of the BPE tokenization algorithm, no normalization of the text was performed in order to have the most general model possible. In all cases, we observed that adding unicode normalization such as NFKC did not reduce the fertility by more than 0.8% on all the languages considered but came at the cost of making the model less general; for example, causing 2² and 2² to be encoded in the same way.

**Pre-tokenizer** Our pre-tokenization has two goals: producing a first division of the text (usually using whitespaces and punctuation) and restricting the maximum length of sequences of tokens produced by the BPE algorithm. The pre-tokenization rule used was the following regex: “\(?[^\(\]\[\]{}\.,\s]+\)\+” which splits words apart while preserving all the characters and in particular the sequences of spaces and line breaks that are crucial for programming languages. We do not use English-centric splits common in other tokenizers (e.g. splitting around ‘nt’ or ‘ll’). We also didn’t use splits on numbers and digits, which caused issues in Arabic and code.

### 3.4 Engineering

#### 3.4.1 Hardware

The model was trained on Jean Zay, a French government-funded supercomputer owned by GENCI and operated at IDRIS, the national computing center for the French National Center for Scientific Research (CNRS). Training BLOOM took about 3.5 months to complete and consumed 1,082,990 compute hours. Training was conducted on 48 nodes, each having 8 NVIDIA A100 80GB GPUs (a total of 384 GPUs); due to possible hardware failures during training, we also maintained a reserve of 4 spare nodes. The nodes were equipped with 2x AMD EPYC 7543 32-Core CPUs and 512 GB of RAM, while the storage was handled by mix of full flash and hard disk drives using a SpectrumScale (GPFS) parallel file system shared between all nodes and users of the supercomputer. 4 NVLink GPU-to-GPU interconnects per node enabled intra-node communications while 4 Omni-Path 100 Gbps links per node, arranged in an enhanced hypercube 8D global topology, were used for inter-node communications.

---

18. [github.com/bigscience-workshop/bs-tokenizers](https://github.com/bigscience-workshop/bs-tokenizers)
3.4.2 Framework

BLOOM was trained using Megatron-DeepSpeed\(^{20}\) (Smith et al., 2022), a framework for large-scale distributed training. It consists of two parts: Megatron-LM\(^{21}\) (Shoeybi et al., 2019) provides the Transformer implementation, tensor parallelism, and data loading primitives, whereas DeepSpeed\(^{22}\) (Rasley et al., 2020) provides the ZeRO optimizer, model pipelining, and general distributed training components. This framework allows us to train efficiently with 3D parallelism (Narayanan et al., 2021, shown in Figure 6), a fusion of three complementary approaches to distributed training. These approaches are described below:

![Diagram of 3D parallelism](image)

**Figure 6:** DP+PP+TP combination leads to 3D parallelism.

**Data parallelism (DP)** replicates the model multiple times, with each replica placed on a different device and fed a slice of the data. The processing is done in parallel and all model replicas are synchronized at the end of each training step.

**Tensor parallelism (TP)** partitions individual layers of the model across multiple devices. This way, instead of having the whole activation or gradient tensor reside on a single GPU, we place shards of this tensor on separate GPUs. This technique is sometimes called horizontal parallelism or intra-layer model parallelism.

**Pipeline parallelism (PP)** splits up the model’s layers across multiple GPUs, so that only a fraction of the layers of the model are placed on each GPU. This is sometimes called vertical parallelism.

Finally, the Zero Redundancy Optimizer (ZeRO; Rajbhandari et al., 2020) allows different processes to only hold a fraction of data (parameters, gradients, and optimizer states).

---

20. github.com/bigscience-workshop/Megatron-DeepSpeed
21. github.com/NVIDIA/Megatron-LM
22. github.com/microsoft/DeepSpeed
required for a training step. We used ZeRO stage 1, meaning that only the optimizer states are sharded in this manner.

The four components described above are combined together to allow scaling to hundreds of GPUs with extremely high GPU utilization. We were able to achieve 156 TFLOPs in our fastest configuration with A100 GPUs, attaining our objective of half of the theoretical peak performance of 312 TFLOPs (in float32 or bfloat16).

3.4.3 Floating Point Format

In earlier experiments with 104B-parameter models on NVIDIA V100 GPUs, we observed numerical instabilities that caused irreversible training divergences. We hypothesize that these instabilities stem from our initial use of IEEE float16 — a 16-bit floating point format with a very limited dynamic range that can cause overflows. The NVIDIA A100 GPUs that we ultimately had access to support the bfloat16 format (Wang and Kanwar, 2019; Kalamkar et al., 2019), which has the same dynamic range as float32. On the other hand, bfloat16 still has much lower precision, which motivated our use of mixed-precision training (Micikevicius et al., 2018). This technique performs certain precision-sensitive operations such as gradient accumulation and softmax in float32 precision and the rest of operations in lower precision, allowing us to achieve a balance of high performance and training stability. Ultimately, we performed final training in bfloat16 mixed precision, which proved to solve the instability problem (in line with previous observation by Smith et al., 2022).

3.4.4 Fused CUDA Kernels

In general, GPUs cannot retrieve data to perform computations on and perform these computations at the same time. Moreover, the compute performance of modern GPUs is much higher than the speed of memory transfer required for every operation (often called a kernel in GPU programming). Kernel fusion (Wu et al., 2012) is an approach for optimizing GPU-based computations by performing several consecutive operations in only one kernel call. This approach offers a way to minimize data transfers: intermediary results stay in the GPU register instead of being copied into VRAM, saving overhead.

We used several custom fused CUDA kernels provided by Megatron-LM. First, we used an optimized kernel to perform LayerNorm, as well as kernels to fuse various combinations of the scaling, masking, and softmax operations. The addition of a bias term is also fused with the GeLU activation using the JIT functionality of PyTorch. As an example consequence of the use of fused kernels, adding the bias term in the GeLU operation adds no additional time, as the operation is memory-bound: the additional computation is negligible compared to data transfers between GPU VRAM and registers, so fusing both operations essentially halves their runtime.

3.4.5 Additional Challenges

Scaling to 384 GPUs required two final changes: disabling asynchronous CUDA kernel launches (for ease of debugging and to prevent deadlocks) and splitting parameter groups into smaller subgroups (to avoid excessive CPU memory allocations).
During training, we faced issues with hardware failures: on average, 1–2 GPU failures occurred each week. As backup nodes were available and automatically used, and check-points were saved every three hours, this did not affect training throughput significantly. A PyTorch deadlock bug in the data loader and disk space issues led to 5–10h downtimes. Given the relative sparsity of engineering issues, and since there was only one loss spike, which the model swiftly recovered from, human intervention was less necessary than in comparable projects (Zhang et al., 2022). Full details of our experience with training BLOOM and a detailed report of all issues we faced are publicly available.\textsuperscript{23}

### 3.5 Training

<table>
<thead>
<tr>
<th>Hyperparameter (↓)</th>
<th>BLOOM-560M</th>
<th>BLOOM-1.1B</th>
<th>BLOOM-1.7B</th>
<th>BLOOM-3B</th>
<th>BLOOM-7.1B</th>
<th>BLOOM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
<td>559M</td>
<td>1,065M</td>
<td>1,722M</td>
<td>3,003M</td>
<td>7,069M</td>
<td>176,247M</td>
</tr>
<tr>
<td>Precision</td>
<td>float16</td>
<td></td>
<td>bfloat16</td>
<td></td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>Layers</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>30</td>
<td>30</td>
<td>14336</td>
</tr>
<tr>
<td>Hidden dim.</td>
<td>1024</td>
<td>1536</td>
<td>2048</td>
<td>2560</td>
<td>4096</td>
<td>112</td>
</tr>
<tr>
<td>Attention heads</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>32</td>
<td>250,680</td>
</tr>
<tr>
<td>Vocab size</td>
<td>250,680</td>
<td>250,680</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence length</td>
<td>2048</td>
<td>2048</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activation</td>
<td>GELU</td>
<td>GELU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position emb.</td>
<td>A1ibi</td>
<td>A1ibi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tied emb.</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Pretraining hyperparameters</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Batch Size</td>
</tr>
<tr>
<td>Learning rate</td>
</tr>
<tr>
<td>Total tokens</td>
</tr>
<tr>
<td>Warmup tokens</td>
</tr>
<tr>
<td>Decay tokens</td>
</tr>
<tr>
<td>Decay style</td>
</tr>
<tr>
<td>Min. learning rate</td>
</tr>
<tr>
<td>Weight decay</td>
</tr>
<tr>
<td>Gradient clipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Multitask finetuning hyperparameters</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Batch Size</td>
</tr>
<tr>
<td>Learning rate</td>
</tr>
<tr>
<td>Total tokens</td>
</tr>
<tr>
<td>Warmup tokens</td>
</tr>
<tr>
<td>Decay style</td>
</tr>
<tr>
<td>Weight decay</td>
</tr>
</tbody>
</table>

Table 3: BLOOM & BLOOMZ Training Hyperparameters.

\textsuperscript{23} github.com/bigscience-workshop/bigscience/blob/master/train/tr11-176B-ml/chronicles.md
Pretrained Models  We train six size variants of BLOOM with respective hyperparameters detailed in Table 3. Architecture and training hyperparameters come from our experimental results (Le Scao et al., 2022) and prior work on training large language models (Brown et al., 2020; Kaplan et al., 2020). Model depth and width for the non-176B models roughly follow previous literature (Brown et al., 2020; Zhang et al., 2022), deviating for 3B and 7.1B in order only to fit the models more easily on our training setup. Embedding parameter sizes are larger for BLOOM owing to the larger multilingual vocabulary, but scaling literature discounts embedding operations (Kaplan et al., 2020). During the development process at the 104B parameters scale, we experimented with different values of Adam $\beta$ parameters, weight decay and gradient clipping to target stability, but did not find it helpful. For all models, we use a cosine learning rate decay schedule (Loshchilov and Hutter, 2016) over 410B tokens, taken as an upper bound for the length of training if compute permitted, and warmup for 375M tokens. We use weight decay, gradient clipping, and no dropout. The ROOTS dataset contains around 341 billion tokens of text, so we aimed to train all models for the equivalent amount of tokens. However, in light of revised scaling laws published during training (Hoffmann et al., 2022), we decided to train the large models for an additional 25 billion tokens on repeated data. As warmup tokens + decay tokens were larger than the total number of tokens, the end of learning rate decay was never reached.

Multitask Finetuning  Finetuned BLOOMZ models (Muennighoff et al., 2022b) maintain the same architecture hyperparameters as BLOOM models. The finetuning hyperparameters are loosely based on T0 (Sanh et al., 2022) and FLAN (Wei et al., 2021). Learning rates are determined by doubling the minimum learning rate of the respective pretrained model and then rounding. Global batch sizes are multiplied by four for small variants to increase throughput. While the models are finetuned for 13 billion tokens, the best checkpoint is chosen according to a separate validation set. We found performance to plateau after 1 – 6 billion tokens of finetuning.

Contrastive Finetuning  We also perform contrastive finetuning of the 1.3 and 7.1 billion parameter BLOOM models using the SGPT Bi-Encoder recipe (Muennighoff, 2022) to train models that produce high-quality text embeddings. We created SGPT-BLOOM-7.1B-msmarco$^{24}$ geared towards multilingual information retrieval and SGPT-BLOOM-1.7B-nli$^{25}$ for multilingual semantic textual similarity (STS). However, recent benchmarking has found these models to also generalize to various other embedding tasks, such as bitext mining, reranking or feature extraction for downstream classification (Muennighoff et al., 2022a).

3.5.1 Carbon Footprint

While most attempts to estimate the carbon footprint of language models have shed light on the emissions produced due to energy consumed during model training (e.g. Patterson et al., 2021; Strubell et al., 2019), other sources of emissions are also important to consider. In our efforts to estimate the carbon emissions of BLOOM, we were inspired by the Life Cycle Assessment (LCA) approach (Klöpffer, 1997) and aimed to consider aspects such as

\[24. \text{hf.co/bigscience/sgpt-bloom-7b1-msmarco}\]
\[25. \text{hf.co/bigscience-data/sgpt-bloom-1b7-nli}\]
the emissions of equipment manufacturing, intermediate model training, and deployment. According to our estimates, the carbon emissions from BLOOM training add up to approximately 81 tons of CO$_2$eq, of which 14% were generated by the equipment manufacturing process (11 tons), 30% by the energy consumed during training (25 tons) and 55% by idle consumption of the equipment and computing cluster used for training (45 tons).

<table>
<thead>
<tr>
<th>Model name</th>
<th>Number of parameters</th>
<th>Power consumption</th>
<th>CO$_2$eq emissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3</td>
<td>175B</td>
<td>1,287 MWh</td>
<td>502 tons</td>
</tr>
<tr>
<td>Gopher</td>
<td>280B</td>
<td>1,066 MWh</td>
<td>352 tons</td>
</tr>
<tr>
<td>OPT</td>
<td>175B</td>
<td>324 MWh</td>
<td>70 tons</td>
</tr>
<tr>
<td>BLOOM</td>
<td>176B</td>
<td>433 MWh</td>
<td>25 tons</td>
</tr>
</tbody>
</table>

Table 4: Comparison of carbon emissions between BLOOM and similar LLMs. Numbers in *italics* have been inferred based on data provided in the papers describing the models.

Comparing the carbon emissions of BLOOM training to other similar models (see Table 4), reveals that while the energy consumption of BLOOM is slightly higher than OPT (Zhang et al., 2022) (433 Mwh compared to OPT’s 324 MWh), its emissions are approximately 2/3 less (25 tons versus 70 tons). This is thanks to the low carbon intensity of the energy grid used for training BLOOM, which emits 57 gCO$_2$eq/kWh, compared to 231 gCO$_2$eq/kWh for the grid used for OPT training. Specifically, France’s national energy grid (which is used by Jean Zay) is largely powered by nuclear energy, which is low-carbon compared to grids powered by energy sources such as coal and natural gas. While the sustainability of nuclear energy is debated, it is one of the least carbon-intensive sources of energy that is currently available. Both BLOOM and OPT incurred significantly less carbon emissions than GPT-3 (as reported by (Patterson et al., 2021)), which can be attributed to several factors including more efficient hardware as well as less carbon-intensive energy sources.

We also pursued further exploration of the carbon footprint of (1) the computation carried out on Jean Zay within the scope of the Big Science workshop, and (2) running the BLOOM model API in real time. In terms of the footprint of the totality of the computation, we estimate that the final BLOOM training represents approximately 37% of the overall emissions, with other processes such as intermediate training runs and model evaluation adding up to the other 63%. This is slightly less than the estimate made by the authors of the OPT paper, who stated that the total carbon footprint of their model is roughly 2 times higher due to experimentation, baselines and ablation (Zhang et al., 2022). Our ongoing exploration of the carbon emissions of the BLOOM API have estimated that the real-time deployment of the model on a GCP instance with 16 GPUs running in the *us-central1* region results in approximately 20 kg of CO$_2$eq emitted per day of deployment (or 0.83 kg per hour). This figure is not representative of all deployment use-cases, and will vary depending on the hardware used as well as the specifics of model implementation (e.g. whether batching is used) and the number of requests the model receives. Further information regarding BLOOM’s carbon footprint can be found in Luccioni et al. (2022).
3.6 Release

Openness has been central to the development of BLOOM and we wanted to ensure it is easily available for the community to use. As such, we worked on producing documentation as a Model Card (Mitchell et al., 2019) and a new license addressing specific goals of the project.

Model Card  Following best practices for releasing machine learning models, the BLOOM model has been released along with a detailed Model Card\(^26\) (Mitchell et al., 2019) describing its technical specifications, details on training, intended-use, out-of-scope uses as well as the model’s limitations. Participants across working groups worked together to produce the final Model Card and similar cards for each checkpoint. The work was collaborative, primarily composed “live” by thinking through and discussing each section, then further dividing into subsections based on the categorizations and distinctions participants naturally ended up creating throughout discussions.

Licensing  Being mindful of the potentially harmful use-cases that BLOOM could enable, we chose to strike a balance between unrestricted open-access and responsible-use by including behavioral-use clauses (Contractor et al., 2022) to limit the application of the model towards potentially harmful use-cases. Such clauses are routinely being included in a growing class of “Responsible AI Licenses (RAIL)”\(^27\) that the community has been adopting when releasing their models.\(^28\) A distinguishing aspect of the RAIL license developed for BLOOM is that it separates licensing of the “source code” and “model”, as referenced by its trained parameters. It further includes detailed definitions of “use” and “derived works” of the model to ensure that anticipated downstream use by prompting, finetuning, distillation, use of logits and probability distributions are explicitly identified. The license contains 13 behavioral-use restrictions that have been identified based on the intended uses and limitations described in the BLOOM Model Card, as well as the BigScience ethical charter. The license offers the model at no charge and users are free to use the model as long as they comply with the terms (including usage restrictions). The source code for BLOOM has been made available under an Apache 2.0 open source license.

4. Evaluation

Our evaluations focus on zero-shot and few-shot settings. Our goal is to present an accurate picture of how BLOOM compares to existing LLMs in settings that most realistically reflect the way the models are likely to be used in practice. Because of the scale of these models, prompt-based adaptation and few-shot “in-context learning” are currently more common than finetuning. Thus, we report results on a range of tasks - SuperGLUE 4.2, machine translation 4.3, summarization 4.4 - and languages in zero-shot and one-shot prompt-based settings, as well as after multitask finetuning (Section 4.7). We also perform code generation 4.5, use BLOOM-derived text embeddings for representation tasks 4.8 and interpret BLOOM’s generalization abilities from the perspective of multilingual probing (Section 4.9).

\(^{26}\) hf.co/bigscience/bloom
\(^{27}\) licenses.ai
\(^{28}\) the-turing-way.netlify.app/reproducible-research/licensing/licensing-ml.html
4.1 Experimental Design

4.1.1 Prompts

Based on recent research on the impact of prompting on language model performance, we decided to build a language model evaluation suite that allowed us to vary both the basic task data as well as the prompting that is used to contextualize the task. Our prompts were developed prior to BLOOM’s release, and did not undergo any a priori refinement using models. That is, the prompts we use in our evaluation are ones that humans believed were a reasonable way to solicit the desired task behavior from a language model. Our goal for designing prompts in this way is to simulate realistic zero-shot or one-shot results that a new user could expect from BLOOM. This is in contrast to presenting best-case performances that might result from multiple rounds of trial-and-error on prompt design. We choose to report the former because the latter is harder to reproduce systematically, is arguably a less representative picture of how the model works in the average setting, and is not representative of true zero-shot learning where no labeled data is available.

We generate multiple prompts per task using promptsource (Bach et al., 2022). We follow the procedure used by Sanh et al. (2022), in which prompt generation is crowdsourced, and thus we see substantial variety in length and style across prompts. To improve quality and clarity, multiple peer reviews were performed on each prompt for artifacts and consistency.

Table 5 shows examples of the resulting prompts used for the WMT’14 task. We also generate prompts for many tasks that are not included in this paper due to resource constraints. All of our prompts for all tasks (both those analyzed in the paper and those not yet analyzed) are publicly available.

<table>
<thead>
<tr>
<th>Prompt name</th>
<th>Prompt</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_good_translation-source+target</td>
<td>Given the following source text: [source sentence], a good L2 translation is: [target sentence]</td>
<td></td>
</tr>
<tr>
<td>gpt3-target</td>
<td>What is the L2 translation of the sentence: [source sentence]? [target sentence]</td>
<td></td>
</tr>
<tr>
<td>version-target</td>
<td>if the original version says [source sentence]; then the L2 version should say: [target sentence]</td>
<td></td>
</tr>
<tr>
<td>xglm-source+target</td>
<td>L1: [source sentence] = L2: [target sentence]</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Four prompts for the WMT’14 dataset (Bojar et al., 2014) for MT evaluation. Above, “L1” and “L2” are replaced with language names (e.g. “Bengali” and “Russian”).

4.1.2 Infrastructure

Our framework extends EleutherAI’s Language Model Evaluation Harness (Gao et al., 2021) by integrating it with the promptsource (Bach et al., 2022) library described in Section 3.1.4. We release our Prompted Language Model Evaluation Harness as an open source library for people to use. We use this framework in order to run the experiments and aggregate results.

29. github.com/bigscience-workshop/promptsource/tree/eval-hackathon
4.1.3 Datasets

**SuperGLUE**  We use a subset of the SuperGLUE (Wang et al., 2019) evaluation suite of classification tasks, specifically: Ax-b, Ax-g, BoolQ, CB, WiC, WSC, and RTE tasks. We excluded the remaining tasks because they require an order of magnitude more compute to run than all of these tasks we consider combined. These tasks are English-only, and are thus included to facilitate comparison with prior work, which has primarily focused on English-only models. We also note that performance on these tasks has not yet been widely reported using zero- and one-shot prompt-based setting. T0 (Sanh et al., 2022) is the first exception, but that model is instruction-tuned and thus not directly comparable to models like BLOOM and OPT. For each task, we select a random sample of five prompts from promptsource and evaluate all models on that set of prompts. As with other prompting tasks in Evaluation Harness (Gao et al., 2021), the prediction of a model for a given prompt is measured using the maximum log likelihood among a set of specified candidate label strings associated with the prompt.

**Machine Translation (MT)**  We evaluate BLOOM on three datasets (using ISO-639-1 codes to refer to languages): WMT14 en↔fr and en↔hi (Bojar et al., 2014), Flores-101 (Goyal et al., 2022) and DiaBLa (Bawden et al., 2020). We evaluate using the sacrebleu (Post, 2018) implementation of BLEU (Papineni et al., 2002), using default tokenisation for WMT and DiaBLa and spm-flores-101 for Flores.\(^{30}\) We use greedy decoding with generation proceeding until the EOS token, or additionally \n###\n for the 1-shot case. The maximum generation length was set per dataset to be in line with what is typically used in the literature; specifically, 64 tokens for WMT14 and 512 tokens for Flores-101 and DiaBLa. Task-specific experimental design details are below.

**Summarization**  We evaluate summarization on the WikiLingua (Ladhak et al., 2020) dataset. WikiLingua is a multilingual summarization dataset comprising WikiHow article and step-by-step summary pairs. Pairs are aligned across multiple languages, with translation of source and summary further reviewed by an international translation team. One-shot conditional natural language generation has typically not been reported by models with size comparable to BLOOM. PaLM (Chowdhery et al., 2022) is the first exception, and reports scores on WikiLingua; however, only the model’s ability to summarize in English was examined (-> en). By contrast, we opted to test BLOOM’s inherent multilingual ability by assessing the abstractive summarization in the source language (e.g. vi -> vi). We focus on the nine languages (Arabic, English, Spanish, French, Hindi, Indonesian, Portuguese, Vietnamese and Chinese) which were amongst those targeted as part of the BigScience effort.

Natural language generation is notoriously challenging to evaluate, with multilingual generation compounding this challenge due to a lack of metric support. Following the suggestions by Gehrmann et al. (2022b), we report ROUGE-2, ROUGE-L (Lin, 2004),\(^{31}\) and Levenshtein distance. One important modification to ROUGE is using the SentencePiece tokenizer (Kudo and Richardson, 2018) built from the Flores-101 dataset (Goyal et al.,

\(^{30}\) BLEU+case:mixed+numrefs.1+smooth.exp+{13a,tok:spm-flores}+version:2.2.1

\(^{31}\) For ROUGE, we used the Python implementation at github.com/google-research/google-research/rouge, commit f935042.
A naive approach would use a tokenizer based on English, but using a multilingual tokenizer improves the capacity to measure the fidelity of multilingual generations. To minimize inference time of the model we use the subsamples from the updated GEM benchmark (Gehrmann et al., 2022a) (3000 uniformly sampled test examples). The authors note that there is minimal difference when comparing model performance between the subsamples and the full test sets. For decoding and generation, we use the same procedure as described above for MT.

4.1.4 Baseline Models

We use the following baseline models where appropriate (e.g. in settings where they support the language of the evaluation dataset):

- **mGPT** (Shliazhko et al., 2022), GPT-style models trained on 60 languages from Wikipedia and Common Crawl
- **GPT-Neo** (Black et al.), **GPT-J-6B** (Wang and Komatsuzaki, 2021), and **GPT-NeoX** (Black et al., 2022), a family of GPT-style models trained on The Pile (Gao et al., 2020)
- **T0** (Sanh et al., 2022), a variant of T5 (Raffel et al., 2020) that underwent multitask prompted finetuning on datasets from P3 (Bach et al., 2022)
- **OPT** (Zhang et al., 2022), a family of GPT-style model trained on a mixture of datasets including those from RoBERTa Liu et al. (2019) and The Pile (Gao et al., 2020)
- **XGLM** (Lin et al., 2021), a GPT-style multilingual model trained on a variant of CC100 (Conneau et al., 2020)
- **M2M** (Fan et al., 2021), a massively multilingual model trained to translate between 100 languages
- **AlexaTM** (Soltan et al., 2022), an encoder-decoder model trained on a mixture of masked and causal language modeling on data from Wikipedia and mC4 (Xue et al., 2021)
- **mTk-Instruct** (Wang et al., 2022b), a variant of T5 that underwent multitask prompted finetuning on datasets from Super-NaturalInstructions
- **Codex** (Chen et al., 2021), a family of GPT models finetuned on code from GitHub
- **GPT-fr** (Simoulin and Crabbé, 2021), a GPT-style model trained on French text

4.2 SuperGLUE

Figure 7 shows zero- and one-shot performance on SuperGLUE. In both settings, on entailment tasks (BoolQ and CB), performance is well above random chance for BLOOM, T0, OPT, and GPT-J. On other tasks, while the best prompts do better, the average performance across prompts hovers around chance, suggesting that the success of individual
prompts is primarily statistical variation. There is some signal for BLOOM in the diagnostic (Ax-b and Ax-g) datasets. The exception is the T0 model, which shows strong performance. However, this model is finetuned in the multitask setting (similar to BLOOMZ, see Section 4.7) in order to improve performance in zero-shot prompting settings, and thus is not directly comparable to the other models shown here.

As models go from zero-shot to one-shot, variability is reduced across all prompts and models and performance slightly and inconsistently increases. Notably, BLOOM sees more of an increase in performance than comparable models when going from zero-shot to one-shot, as it is generally behind OPT in the zero-shot setting but matches or improves on it in the one-shot setting, even though it has only partly been trained on English. This may be because a multilingual language model gains more certainty in the language of input and output with a longer context.

![Figure 7: Performance of various LLMs on subset of tasks from SuperGLUE benchmark in zero- and one-shot prompt-based setting.](image)

We perform an additional analysis comparing BLOOM models across model sizes. As a baseline, we also measure the average one-shot accuracy of OPT models of similar sizes (350M parameters to 175B parameters). Figure 8 shows the accuracy of each prompt on each task across model scales. Both OPT and BLOOM model families improve very slightly with scale, with only models over 2 billion parameters showing signal, and there is no consistent difference between families across all tasks. In the 1-shot setting, BLOOM-176B is ahead of OPT-175B on Ax-b, CB, WSC and WiC, and matches it on the other tasks, suggesting that multilinguality does not limit performance of BLOOM on English-only tasks in the zero-shot setting.

32. We do not evaluate OPT-66B because of the lack of a similarly-sized BLOOM model.
4.3 Machine Translation

In addition to the results presented here, a more detailed analysis of BLOOM’s MT quality can be found in (Bawden and Yvon, 2023).

4.3.1 WMT

WMT results for BLOOM-176B in the zero-shot and 1-shot setting are given in Table 6. The best prompts tend to be the more verbose ones; the “version-target” prompt is consistently better and the “gpt3-target” and “xglm-source+target” prompts have very poor performance, especially for zero-shot. In the one-shot setting, BLOOM can, with the right prompt, perform competent translation, although it is behind dedicated (supervised) models such as M2M-100 (43.8 BLEU for English→French and 40.4 for French→English, compared to 34.2 and 35.4 BLEU for BLOOM). The two major problems observed, particularly in the zero-shot setting, are (i) over-generation and (ii) not producing the correct language (an
obvious prerequisite for a good translation). Both of these aspects are greatly improved as the number of few-shot examples is increased.

<table>
<thead>
<tr>
<th>Prompt</th>
<th>en → fr</th>
<th>fr → en</th>
<th>en → hi</th>
<th>hi → en</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shots</td>
<td>0 1</td>
<td>0 1</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>a_good_translation-source+target</td>
<td>15.38 36.39</td>
<td>14.15 36.56</td>
<td>1.90 14.49</td>
<td>10.19 24.60</td>
</tr>
<tr>
<td>gpt3-target</td>
<td>7.90 32.55</td>
<td>12.73 33.14</td>
<td>0.26 6.51</td>
<td>0.66 9.98</td>
</tr>
<tr>
<td>version-target</td>
<td>21.96 34.22</td>
<td>26.79 35.42</td>
<td>1.96 13.95</td>
<td>11.48 25.80</td>
</tr>
<tr>
<td>xglm-source+target</td>
<td>14.91 27.83</td>
<td>15.52 34.51</td>
<td>6.80 13.62</td>
<td>12.05 25.04</td>
</tr>
</tbody>
</table>

Table 6: WMT’14 zero- and one-shot results (BLEU) for BLOOM-176B. The prompts used are described in Table 5.

4.3.2 DiaBLA

<table>
<thead>
<tr>
<th>1-shot context</th>
<th>Truncate</th>
<th>en→fr</th>
<th>fr→en</th>
<th>en→fr</th>
<th>fr→en</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand.</td>
<td>×</td>
<td>5.7</td>
<td>0.342</td>
<td>12.1</td>
<td>0.614</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td><strong>37.6</strong></td>
<td><strong>0.634</strong></td>
<td>41.4</td>
<td><strong>0.757</strong></td>
</tr>
<tr>
<td>Prev.</td>
<td>×</td>
<td>6.1</td>
<td>0.328</td>
<td>12.3</td>
<td>0.617</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td><strong>38.5</strong></td>
<td>0.614</td>
<td><strong>41.6</strong></td>
<td>0.751</td>
</tr>
</tbody>
</table>

Table 7: DiaBLA 1-shot results (BLEU) for the “xglm-source+target” prompt when using the previous or a random sentence as the 1-shot example (with and without truncation of outputs). In bold the best results for each direction.

Table 7 shows results testing the use of linguistic context with DiaBLA, a parallel dataset of informal bilingual dialogues. In a 1-shot context and using the “xglm-source+target” prompt, we compare the effect of using a random test set example as the 1-shot example versus using the previous dialogue utterance. In light of the overgeneration issues seen and in order to evaluate the quality of the prediction independently of overgeneration, we report results for both original outputs and after applying a custom truncation function. The automatic results are inconclusive, with little difference between scores (BLEU scores are higher for previous context but COMET scores are lower). Despite these results, there is evidence in the predictions themselves that the model is able to use the context of the 1-shot example to make translation choices. See (Bawden and Yvon, 2023) for examples and further analysis.
### 4.3.3 Flores

In the 1-shot setting, we test several language directions in the Flores-101 (Goyal et al., 2022) devtest set using the “xglm-source+target” prompt (Lin et al., 2021). The 1-shot example is randomly taken from the dev set. We separate out results for low-resource language pairs (Table 8a), between related languages of the Romance language family (Table 8b), high-resource language pairs (Table 8c) and high-to-mid-resource language pairs (Table 8d).

33. The truncation rule is specific to the “xglm-source+target” prompt and the fact that overgeneration consists of repeating the prompt pattern. Anything after a first newline or the regular expression pattern = .+?: is discarded.

#### Table 8: 1-shot MT results (spBLEU) on the Flores-101 devtest set.

<table>
<thead>
<tr>
<th>Src ↓</th>
<th>Trg →</th>
<th>en</th>
<th>bn</th>
<th>hi</th>
<th>sw</th>
<th>yo</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>BLOOM</td>
<td>24.6</td>
<td>27.2</td>
<td>20.5</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>23.0</td>
<td>28.1</td>
<td>26.9</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>bn</td>
<td>BLOOM</td>
<td>29.9</td>
<td>16.3</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>22.9</td>
<td>21.8</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>hi</td>
<td>BLOOM</td>
<td>35.1</td>
<td>23.8</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>27.9</td>
<td>21.8</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>sw</td>
<td>BLOOM</td>
<td>37.4</td>
<td>-</td>
<td>0.9</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>30.4</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>yo</td>
<td>BLOOM</td>
<td>4.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>4.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

#### (a) Low-resource languages

<table>
<thead>
<tr>
<th>Src ↓</th>
<th>Trg →</th>
<th>ar</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>zh</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>BLOOM</td>
<td>40.3</td>
<td>23.3</td>
<td>33.1</td>
<td>17.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>25.5</td>
<td>16.7</td>
<td>25.7</td>
<td>13.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AlexaTM</td>
<td>41.8</td>
<td>23.2</td>
<td>35.5</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>en</td>
<td>BLOOM</td>
<td>28.2</td>
<td>29.4</td>
<td>45.0</td>
<td>26.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>17.9</td>
<td>25.6</td>
<td>42.0</td>
<td>19.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AlexaTM</td>
<td>32.0</td>
<td>31.0</td>
<td>50.7</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>es</td>
<td>BLOOM</td>
<td>18.8</td>
<td>32.7</td>
<td>-</td>
<td>24.8</td>
<td>20.9</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>12.1</td>
<td>25.1</td>
<td>-</td>
<td>29.3</td>
<td>14.9</td>
</tr>
<tr>
<td></td>
<td>AlexaTM</td>
<td>20.8</td>
<td>34.6</td>
<td>-</td>
<td>33.4</td>
<td>-</td>
</tr>
<tr>
<td>fr</td>
<td>BLOOM</td>
<td>23.4</td>
<td>45.6</td>
<td>27.5</td>
<td>-</td>
<td>23.2</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>15.4</td>
<td>37.2</td>
<td>25.6</td>
<td>-</td>
<td>17.6</td>
</tr>
<tr>
<td></td>
<td>AlexaTM</td>
<td>24.7</td>
<td>47.1</td>
<td>26.3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>zh</td>
<td>BLOOM</td>
<td>15.0</td>
<td>30.5</td>
<td>20.5</td>
<td>26.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>11.55</td>
<td>20.9</td>
<td>16.9</td>
<td>24.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>AlexaTM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### (b) Romance languages

<table>
<thead>
<tr>
<th>Src ↓</th>
<th>Trg →</th>
<th>ca</th>
<th>es</th>
<th>fr</th>
<th>gl</th>
<th>it</th>
<th>pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca</td>
<td>BLOOM</td>
<td>28.9</td>
<td>33.8</td>
<td>19.2</td>
<td>19.8</td>
<td>23.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>25.2</td>
<td>35.1</td>
<td>33.4</td>
<td>25.5</td>
<td>35.2</td>
<td></td>
</tr>
<tr>
<td>es</td>
<td>BLOOM</td>
<td>31.2</td>
<td>-</td>
<td>24.8</td>
<td>23.3</td>
<td>16.5</td>
<td>29.1</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>23.1</td>
<td>-</td>
<td>29.3</td>
<td>27.5</td>
<td>23.9</td>
<td>28.1</td>
</tr>
<tr>
<td>fr</td>
<td>BLOOM</td>
<td>37.2</td>
<td>27.5</td>
<td>-</td>
<td>24.9</td>
<td>24.0</td>
<td>38.9</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>28.7</td>
<td>25.6</td>
<td>-</td>
<td>32.8</td>
<td>28.6</td>
<td>37.8</td>
</tr>
<tr>
<td>gl</td>
<td>BLOOM</td>
<td>37.5</td>
<td>27.1</td>
<td>33.8</td>
<td>-</td>
<td>18.3</td>
<td>32.2</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>30.1</td>
<td>27.6</td>
<td>37.1</td>
<td>-</td>
<td>26.9</td>
<td>34.8</td>
</tr>
<tr>
<td>it</td>
<td>BLOOM</td>
<td>31.0</td>
<td>25.4</td>
<td>31.4</td>
<td>20.2</td>
<td>-</td>
<td>29.2</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>25.2</td>
<td>29.2</td>
<td>34.4</td>
<td>29.2</td>
<td>-</td>
<td>31.5</td>
</tr>
<tr>
<td>pt</td>
<td>BLOOM</td>
<td>39.6</td>
<td>28.1</td>
<td>40.3</td>
<td>27.1</td>
<td>20.1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>30.7</td>
<td>26.9</td>
<td>40.2</td>
<td>33.8</td>
<td>28.1</td>
<td>-</td>
</tr>
</tbody>
</table>

#### (c) High-resource language pairs

<table>
<thead>
<tr>
<th>Src ↓</th>
<th>Trg →</th>
<th>en</th>
<th>fr</th>
<th>hi</th>
<th>id</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>BLOOM</td>
<td>45.0</td>
<td>27.2</td>
<td>39.0</td>
<td>28.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>42.0</td>
<td>28.1</td>
<td>37.3</td>
<td>35.1</td>
<td></td>
</tr>
<tr>
<td>fr</td>
<td>BLOOM</td>
<td>45.6</td>
<td>-</td>
<td>18.5</td>
<td>31.4</td>
<td>32.8</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>37.2</td>
<td>-</td>
<td>22.9</td>
<td>29.1</td>
<td>30.3</td>
</tr>
<tr>
<td>hi</td>
<td>BLOOM</td>
<td>35.1</td>
<td>27.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>27.9</td>
<td>25.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>id</td>
<td>BLOOM</td>
<td>43.2</td>
<td>30.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>33.7</td>
<td>30.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>vi</td>
<td>BLOOM</td>
<td>38.7</td>
<td>26.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M2M</td>
<td>29.5</td>
<td>25.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### (d) High-to-mid-resource language pairs.
Languages are classified as low-, mid- and high-resource depending on their representation in ROOTS. We compare to supervised results from the M2M-100 model (Fan et al., 2021) with 615M parameters, for which scores are computed by Goyal et al. (2022). Additionally, we compare to 32-shot AlexaTM results for high-resource language pairs (Soltan et al., 2022). Results are good across the board for both translation between high-resource languages and from high- to mid-resource languages, suggesting BLOOM’s good multilingual capacity, even across scripts (here between Latin (or extended Latin), Chinese, Arabic and Devanagari scripts). Compared to the supervised M2M-100 model, results are often comparable and sometimes better in this 1-shot setting, and results are comparable in many cases to those of AlexaTM (even though AlexTM results are for 32-shot).

The translation quality for many of the low-resource languages is good, comparable to or even slightly better than the supervised M2M model. However, results are very poor between Swahili and Yoruba, languages that are present but under-represented in BLOOM’s training data (<50k tokens each). This contrasts with the results for translation between Romance (and therefore related) languages, where results are good across the board, including for translation from Galician (glg), a language not included in the training data, but which shares many similarities with the other Romance languages, in particular with Portuguese (por). This however does question BLOOM’s quality on those under-represented low-resource languages included in training.

4.4 Summarization

Figure 9 shows one-shot results for BLOOM models alongside OPT-175B for comparison. Each point represents a per-prompt score. The key takeaways are that BLOOM attains higher performance on multilingual summarization than OPT and that performance increases as the parameter count of the model increases. We suspect this is due to BLOOM’s multilingual-focused training.

As discussed in Section 4.1, we report ROUGE-2 scores for the sake of comparability with prior work, and because there is a lack of alternatives for generation evaluation. However, we qualitatively observe that in many cases, the ROUGE-2 score understates the quality of the summaries generated by the systems.

4.5 Code Generation

The BLOOM pretraining corpus, ROOTS, consists of around 11% of code. In Table 9, we report benchmarking results of BLOOM on HumanEval (Chen et al., 2021). We find the performance of pretrained BLOOM models to be similar to that of the similar-sized GPT models trained on the Pile (Gao et al., 2020). The Pile contains English data and around 13% of code (GitHub + StackExchange), which is similar to the code data sources and proportions in ROOTS. The Codex models, which have solely been finetuned on code, are significantly stronger than other models. Multitask finetuned BLOOMZ models do not improve significantly over BLOOM models. We hypothesize this is due to the finetuning dataset, xP3, not containing significant amounts of pure code completion. Rather, xP3 contains code-related tasks, such as estimating the time complexity of a given Python code snippet. Additional analysis is provided in Muennighoff et al. (2022b).
4.6 HELM benchmark

For completeness, we reproduce here evaluations from the HELM benchmark (Liang et al., 2022), which ran 5-shot evaluations of a variety of language models on English-only tasks. Despite the multilingual training, BLOOM is roughly on par in accuracy with previous-generation English-only models, such as GPT3-davinci v1 and J1-Grande v1, but behind more recent monolingual models such as InstructGPT davinci v2, Turing NLG v2, Anthropic-LM v4-s3, or OPT. Like other large language models of this size, it is not very well calibrated, but quite robust. Finally, on this benchmark, it is one of the best models for fairness, slightly more toxic than average in English, and average for bias.

4.7 Multitask Finetuning

Building on recent work on multitask finetuning (Sanh et al., 2022; Wei et al., 2021; Wang et al., 2022a) we explore using multilingual multitask finetuning to improve the zero-shot performance of the BLOOM model. We conducted multilingual multitask finetuning of BLOOM models using the xP3 corpus outlined in Section 3.1.4. We find that zero-shot performance significantly increases. In Figure 11, we compare the zero-shot performance of pretrained BLOOM and XGLM models with multitask finetuned BLOOMZ, T0 and mTk-Instruct (Wang et al., 2022b). BLOOM and XGLM performances are near the random baselines of 33% for NLI (XNLI) and 50% for coreference resolution (XWinograd) and
<table>
<thead>
<tr>
<th></th>
<th>PASS@k</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k = 1</td>
<td>k = 10</td>
<td>k = 100</td>
</tr>
<tr>
<td>GPT-Neo 1.3B</td>
<td>4.79%</td>
<td>7.47%</td>
<td>16.30%</td>
</tr>
<tr>
<td>GPT-Neo 2.7B</td>
<td>6.41%</td>
<td>11.27%</td>
<td>21.37%</td>
</tr>
<tr>
<td>GPT-J 6B</td>
<td>11.62%</td>
<td>15.74%</td>
<td>27.74%</td>
</tr>
<tr>
<td>GPT-NeoX 20B</td>
<td>15.4%</td>
<td>25.6%</td>
<td>41.2%</td>
</tr>
<tr>
<td>Codex-300M</td>
<td>13.17%</td>
<td>20.37%</td>
<td>36.27%</td>
</tr>
<tr>
<td>Codex-679M</td>
<td>16.22%</td>
<td>25.7%</td>
<td>40.95%</td>
</tr>
<tr>
<td>Codex-2.5B</td>
<td>21.36%</td>
<td>35.42%</td>
<td>59.5%</td>
</tr>
<tr>
<td>Codex-12B</td>
<td>28.81%</td>
<td>46.81%</td>
<td>72.31%</td>
</tr>
<tr>
<td>BLOOM-560M</td>
<td>0.82%</td>
<td>3.02%</td>
<td>5.91%</td>
</tr>
<tr>
<td>BLOOM-1.1B</td>
<td>2.48%</td>
<td>5.93%</td>
<td>9.62%</td>
</tr>
<tr>
<td>BLOOM-1.7B</td>
<td>4.03%</td>
<td>7.45%</td>
<td>12.75%</td>
</tr>
<tr>
<td>BLOOM-3B</td>
<td>6.48%</td>
<td>11.35%</td>
<td>20.43%</td>
</tr>
<tr>
<td>BLOOM-7.1B</td>
<td>7.73%</td>
<td>17.38%</td>
<td>29.47%</td>
</tr>
<tr>
<td>BLOOM</td>
<td>15.52%</td>
<td>32.20%</td>
<td>55.45%</td>
</tr>
<tr>
<td>BLOOMZ-560M</td>
<td>2.18%</td>
<td>4.11%</td>
<td>9.00%</td>
</tr>
<tr>
<td>BLOOMZ-1.1B</td>
<td>2.63%</td>
<td>6.22%</td>
<td>11.68%</td>
</tr>
<tr>
<td>BLOOMZ-1.7B</td>
<td>4.38%</td>
<td>8.73%</td>
<td>16.09%</td>
</tr>
<tr>
<td>BLOOMZ-3B</td>
<td>6.29%</td>
<td>11.94%</td>
<td>19.06%</td>
</tr>
<tr>
<td>BLOOMZ-7.1B</td>
<td>8.06%</td>
<td>15.03%</td>
<td>27.49%</td>
</tr>
<tr>
<td>BLOOMZ</td>
<td>12.06%</td>
<td>26.53%</td>
<td>48.44%</td>
</tr>
</tbody>
</table>

Table 9: Performance on HumanEval (Chen et al., 2021). Non-BLOOM results come from prior work (Chen et al., 2021; Fried et al., 2022). The Codex model is a language model that was finetuned on code, while the GPT models (Black et al.; Wang and Komatsuzaki, 2021; Black et al., 2022) are trained on a mix of code and text like BLOOM.

sentence completion (XCOPA and XStoryCloze). After going through multilingual multitask finetuning (BLOOMZ), zero-shot performance significantly improves on the depicted held-out tasks. Despite also being multitask finetuned, T0 performs badly on the multilingual datasets shown due to it being a monolingual English model. Additional results provided in Muennighoff et al. (2022b), however, show that models finetuned on xP3 also outperform T0 on English datasets when controlling for size and architecture. This is likely due to T0’s finetuning dataset (P3) containing less diverse datasets and prompts than xP3. Multitask finetuning performance has been shown to correlate with the amount of datasets and prompts (Chung et al., 2022).

### 4.8 Embeddings

In Section 3.5, we have outlined the contrastive finetuning procedure for creating SGPT-BLOOM text embedding models. In Table 10, we report benchmarking results on two
multilingual datasets from the Massive Text Embedding Benchmark (MTEB, Muennighoff et al., 2022a). We find that SGPT-BLOOM-7.1B-msmarco provides state-of-the-art performance on several classification and semantic textual similarity splits. However, with 7.1 billion parameters it is an order of magnitude larger than models like the displayed multilingual MiniLM and MPNet. SGPT-BLOOM-1.7B-nli performs significantly worse, likely due to less parameters and its finetuning being shorter (NLI is a much smaller dataset than MS-MARCO). Apart from the BLOOM models, ST5-XL is the largest model with 1.2 billion parameters. However, as an English-only model its performance on non-English

Figure 10: Results for a wide variety of language models on the 5-shot HELM benchmark. Taken from Liang et al. (2022)

36. hf.co/bigscience/gpt-bloom-7b1-msmarco
37. hf.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
38. hf.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
39. hf.co/bigscience/gpt-bloom-1b7-nli
40. hf.co/sentence-transformers/sentence-t5-xl
languages is poor. The languages displayed are part of the BLOOM pretraining corpus. Performance on more languages and datasets can be inspected on the MTEB leaderboard\textsuperscript{41}.

### 4.9 Multilingual Probing

Probing has emerged as a significant evaluation paradigm to analyze and interpret the inner workings of LLMs (Ettinger et al., 2016; Adi et al., 2017; Belinkov et al., 2017; Hupkes et al., 2017).

\textsuperscript{41} hf.co/spaces/mteb/leaderboard
<table>
<thead>
<tr>
<th></th>
<th>ST5-XL</th>
<th>LASER2</th>
<th>MiniLM-L12</th>
<th>MPNet</th>
<th>LaBSE</th>
<th>SGPT-BLOOM-1.7B</th>
<th>SGPT-BLOOM-7.1B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Embedding classification performance on MASSIVE (FitzGerald et al., 2022) scored using accuracy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arabic (ar)</td>
<td>4.18</td>
<td>37.16</td>
<td>51.43</td>
<td>45.14</td>
<td>50.86</td>
<td>54.59</td>
<td><strong>59.25</strong></td>
</tr>
<tr>
<td>Bengali (bn)</td>
<td>2.60</td>
<td>42.51</td>
<td>48.79</td>
<td>35.34</td>
<td>58.22</td>
<td>57.76</td>
<td><strong>61.59</strong></td>
</tr>
<tr>
<td>English (en)</td>
<td><strong>72.09</strong></td>
<td>47.91</td>
<td>69.32</td>
<td>66.84</td>
<td>61.46</td>
<td>66.69</td>
<td><strong>69.67</strong></td>
</tr>
<tr>
<td>Spanish (es)</td>
<td>57.97</td>
<td>45.44</td>
<td>64.43</td>
<td>59.66</td>
<td>58.32</td>
<td>61.77</td>
<td><strong>66.35</strong></td>
</tr>
<tr>
<td>French (fr)</td>
<td>60.99</td>
<td>46.13</td>
<td>64.82</td>
<td>60.25</td>
<td>60.47</td>
<td>64.58</td>
<td><strong>66.95</strong></td>
</tr>
<tr>
<td>Hindi (hi)</td>
<td>3.02</td>
<td>40.20</td>
<td>62.77</td>
<td>58.37</td>
<td>59.40</td>
<td>60.74</td>
<td><strong>63.54</strong></td>
</tr>
<tr>
<td>Indonesian (id)</td>
<td>41.53</td>
<td>45.81</td>
<td><strong>65.43</strong></td>
<td>59.85</td>
<td>61.12</td>
<td>60.07</td>
<td>64.06</td>
</tr>
<tr>
<td>Kannada (kn)</td>
<td>2.79</td>
<td>4.32</td>
<td>50.63</td>
<td>40.98</td>
<td><strong>56.24</strong></td>
<td>48.56</td>
<td><strong>53.54</strong></td>
</tr>
<tr>
<td>Malayalam (ml)</td>
<td>2.98</td>
<td>41.33</td>
<td>54.34</td>
<td>42.41</td>
<td>57.91</td>
<td>55.10</td>
<td><strong>58.27</strong></td>
</tr>
<tr>
<td>Portuguese (pt)</td>
<td>57.95</td>
<td>48.55</td>
<td>64.89</td>
<td>61.27</td>
<td>60.16</td>
<td>62.52</td>
<td><strong>66.69</strong></td>
</tr>
<tr>
<td>Swahili (sw)</td>
<td>30.60</td>
<td>31.89</td>
<td>31.95</td>
<td>29.57</td>
<td><strong>51.62</strong></td>
<td>43.90</td>
<td>49.81</td>
</tr>
<tr>
<td>Tamil (ta)</td>
<td>1.79</td>
<td>29.63</td>
<td>50.17</td>
<td>36.77</td>
<td>55.04</td>
<td>52.66</td>
<td><strong>56.40</strong></td>
</tr>
<tr>
<td>Telugu (te)</td>
<td>2.26</td>
<td>36.03</td>
<td>52.82</td>
<td>40.72</td>
<td><strong>58.32</strong></td>
<td>49.32</td>
<td>54.71</td>
</tr>
<tr>
<td>Urdu (ur)</td>
<td>2.70</td>
<td>26.11</td>
<td>56.37</td>
<td>52.80</td>
<td>56.70</td>
<td>51.00</td>
<td><strong>56.75</strong></td>
</tr>
<tr>
<td>Vietnamese (vi)</td>
<td>21.47</td>
<td>44.33</td>
<td>59.68</td>
<td>56.61</td>
<td>56.67</td>
<td>59.85</td>
<td><strong>64.53</strong></td>
</tr>
</tbody>
</table>

| **Semantic textual similarity on STS22 (Madabushi et al., 2022) scored using spearman correlation of cosine similarities** |
| Arabic (ar)    | 29.60  | 42.57  | 52.19      | 46.20 | 57.67 | 48.64           | **58.67**       |
| English (en)   | 64.32  | 39.76  | 63.06      | 61.72 | **60.97** | 61.45         | **66.13**       |
| Spanish (es)   | 58.16  | 54.92  | 59.91      | 56.56 | 63.18 | 61.81           | **65.41**       |
| French (fr)    | 77.49  | 58.61  | 74.30      | 70.55 | 77.95 | 73.18           | **80.38**       |
| Chinese (zh)   | 33.55  | 49.41  | 61.75      | 58.75 | 63.92 | 58.53           | **66.78**       |

Table 10: Performance of BLOOM models finetuned for sentence embeddings on classification and STS datasets from MTEB (Muennighoff et al., 2022b).

2018; Tenney et al., 2018; Belinkov and Glass, 2019; Teehan et al., 2022), although it comes with certain shortcomings (Belinkov, 2022). Examination of the LLM embeddings can help shed light on the generalizing abilities of the model apart from its training objective loss or downstream task evaluation, which is especially beneficial for examining languages lacking annotated datasets or benchmarks.

**4.9.1 Method**

For interpreting BLOOM’s multilingual generalizing abilities, we utilize the “Universal Probing” framework\(^{42}\) for systematic probing analysis in 104 languages and 80 morphosyntactic features (Serikov et al., 2022). The framework provides SentEval-style (Conneau et al., 2018) probing setup and datasets for each language available in Universal Dependencies (UD; Nivre et al., 2016). We consider the following 17 languages from 7 language families present in BLOOM’s pretraining corpus (Section 3.1) and UD treebanks: Arabic (Afro-Asiatic), Bambara (Mande), Basque (language isolate), Bengali, Catalan, English, French, Hindi, Marathi, Portuguese, Spanish, Urdu (Indo-European), Chinese (Sino-Tibetan), Indonesian (Austronesian), Tamil (Dravidian), Wolof, Yoruba (Niger-Congo). Our setup covers 38 morphosyntactic features in total, which represent language-specific linguistic information. We provide a dataset sample in Table 11.

The probing procedure is conducted as follows. First, we compute \(<s>-pooled representations of the input sentence at each layer of the 1.7B-parameter BLOOM variant

\(^{42}\)github.com/bigscience-workshop/massive-probing-framework
<table>
<thead>
<tr>
<th>Language</th>
<th>Label</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Sing</td>
<td>The <strong>scheme</strong> makes money through sponsorship and advertising.</td>
</tr>
<tr>
<td></td>
<td>Plur</td>
<td>Still, there are <strong>questions</strong> left unanswered.</td>
</tr>
<tr>
<td>Spanish</td>
<td>Sing</td>
<td><strong>Elige</strong> no ir tras un tercer período en el siguiente ciclo de elecciones.</td>
</tr>
<tr>
<td></td>
<td>Plur</td>
<td>Todavía quedan <strong>preguntas</strong> sin responder.</td>
</tr>
</tbody>
</table>

Table 11: Examples of the Number task in English and Spanish. The subject number indicator is highlighted in **bold**. The task is to predict if the sentence includes a singular subject number (upper sentence) and a plural subject number (bottom sentence).

(“BLOOM 1B7”) and BLOOM (with 176B parameters). Second, we train a binary logistic regression classifier to predict a presence of a morphosyntactic feature in the sentence. Logistic regression is chosen due to its higher selectivity as opposed to non-linear probing classifiers (Hewitt and Liang, 2019). We use the original UD training, validation, and test splits here. Third, the probing performance is evaluated by $F_1$ weighted score due to target class imbalance for most probing tasks. The results are averaged across three runs with different random seeds.

**Baselines** We compare the probing performance with random guessing and logistic regression classifiers trained on the following TF-IDF features (Salton and Yang, 1973): word unigrams, character N-grams, BPE$^{43}$ token N-grams, and SentencePiece$^{44}$ (SP; Kudo and Richardson, 2018) token N-grams. We use the N-gram range $\in [1; 4]$ and limit the TF-IDF vocabularies to top-250k features.

**Correlation** We run statistical tests to analyze correlations between the probing performance and linguistic, dataset, and model configuration criteria:

- Language script: the results are divided into two groups by the language script – Latin and others (Devanagari, Tamil, and Arabic). Here, we use the non-parametric test Mann-Whitney U (Mann and Whitney, 1947).
- Language family: the results are divided into 7 groups by the language family. We apply the ANOVA to analyze the variance between the groups.
- Probing and pretraining dataset size: we run the Pearson correlation coefficient test (Pearson, 1895) to compute correlation between the probing performance and these data configuration criteria.
- Effect of the model size: the results are divided into two groups by the BLOOM version. Here, we use the Mann-Whitney U test to see if there is a correlation between the number of parameters and the probing results.
Table 12: Probing performance ($F_1$ averaged by layers) of the BLOOM-based classifiers and count-based baselines. The results are averaged over probing tasks, and three experiment runs within each language. Standard deviation is determined by the results along the language tasks.

<table>
<thead>
<tr>
<th>Language</th>
<th>BLOOM-1B7</th>
<th>BLOOM</th>
<th>Random</th>
<th>TF-IDF (Char)</th>
<th>TF-IDF (Word)</th>
<th>TF-IDF (BPE)</th>
<th>TF-IDF (SP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>0.66 ± 0.27</td>
<td>0.64 ± 0.27</td>
<td>0.49 ± 0.013</td>
<td>0.41 ± 0.44</td>
<td>0.4 ± 0.44</td>
<td>0.41 ± 0.44</td>
<td>0.41 ± 0.44</td>
</tr>
<tr>
<td>Bambara</td>
<td>0.64 ± 0.16</td>
<td>0.59 ± 0.16</td>
<td>0.45 ± 0.1</td>
<td>0.52 ± 0.46</td>
<td>0.45 ± 0.47</td>
<td>0.48 ± 0.49</td>
<td>0.49 ± 0.49</td>
</tr>
<tr>
<td>Basque</td>
<td>0.68 ± 0.19</td>
<td>0.62 ± 0.19</td>
<td>0.49 ± 0.03</td>
<td>0.41 ± 0.43</td>
<td>0.44 ± 0.46</td>
<td>0.48 ± 0.44</td>
<td>0.41 ± 0.46</td>
</tr>
<tr>
<td>Bengali</td>
<td>0.42 ± 0.15</td>
<td>0.45 ± 0.12</td>
<td>0.35 ± 0.23</td>
<td>0.63 ± 0.48</td>
<td>0.37 ± 0.44</td>
<td>0.41 ± 0.32</td>
<td>0.76 ± 0.28</td>
</tr>
<tr>
<td>Catalan</td>
<td>0.65 ± 0.25</td>
<td>0.61 ± 0.26</td>
<td>0.34 ± 0.01</td>
<td>0.24 ± 0.38</td>
<td>0.24 ± 0.39</td>
<td>0.24 ± 0.39</td>
<td>0.24 ± 0.39</td>
</tr>
<tr>
<td>Chinese</td>
<td>0.66 ± 0.25</td>
<td>0.50 ± 0.21</td>
<td>0.55 ± 0.03</td>
<td>0.03 ± 0.05</td>
<td>0.11 ± 0.28</td>
<td>0.04 ± 0.06</td>
<td>0.03 ± 0.05</td>
</tr>
<tr>
<td>English</td>
<td>0.57 ± 0.24</td>
<td>0.57 ± 0.24</td>
<td>0.43 ± 0.03</td>
<td>0.45 ± 0.43</td>
<td>0.46 ± 0.43</td>
<td>0.45 ± 0.43</td>
<td>0.44 ± 0.44</td>
</tr>
<tr>
<td>French</td>
<td>0.61 ± 0.23</td>
<td>0.57 ± 0.22</td>
<td>0.44 ± 0.02</td>
<td>0.32 ± 0.43</td>
<td>0.32 ± 0.43</td>
<td>0.32 ± 0.43</td>
<td>0.33 ± 0.44</td>
</tr>
<tr>
<td>Hindi</td>
<td>0.63 ± 0.23</td>
<td>0.6 ± 0.25</td>
<td>0.48 ± 0.03</td>
<td>0.53 ± 0.46</td>
<td>0.55 ± 0.47</td>
<td>0.53 ± 0.46</td>
<td>0.53 ± 0.46</td>
</tr>
<tr>
<td>Indonesian</td>
<td>0.65 ± 0.27</td>
<td>0.6 ± 0.27</td>
<td>0.48 ± 0.05</td>
<td>0.41 ± 0.46</td>
<td>0.43 ± 0.45</td>
<td>0.41 ± 0.46</td>
<td>0.45 ± 0.45</td>
</tr>
<tr>
<td>Marathi</td>
<td>0.57 ± 0.25</td>
<td>0.48 ± 0.24</td>
<td>0.32 ± 0.09</td>
<td>0.44 ± 0.47</td>
<td>0.46 ± 0.46</td>
<td>0.44 ± 0.47</td>
<td>0.44 ± 0.47</td>
</tr>
<tr>
<td>Portuguese</td>
<td>0.67 ± 0.23</td>
<td>0.63 ± 0.26</td>
<td>0.4 ± 0.03</td>
<td>0.48 ± 0.48</td>
<td>0.49 ± 0.48</td>
<td>0.48 ± 0.48</td>
<td>0.48 ± 0.48</td>
</tr>
<tr>
<td>Spanish</td>
<td>0.66 ± 0.24</td>
<td>0.65 ± 0.24</td>
<td>0.42 ± 0.02</td>
<td>0.35 ± 0.42</td>
<td>0.35 ± 0.44</td>
<td>0.36 ± 0.44</td>
<td>0.36 ± 0.43</td>
</tr>
<tr>
<td>Tamil</td>
<td>0.57 ± 0.25</td>
<td>0.51 ± 0.27</td>
<td>0.43 ± 0.05</td>
<td>0.51 ± 0.44</td>
<td>0.53 ± 0.44</td>
<td>0.5 ± 0.44</td>
<td>0.5 ± 0.44</td>
</tr>
<tr>
<td>Urdu</td>
<td>0.75 ± 0.21</td>
<td>0.70 ± 0.24</td>
<td>0.43 ± 0.02</td>
<td>0.39 ± 0.48</td>
<td>0.39 ± 0.47</td>
<td>0.39 ± 0.48</td>
<td>0.39 ± 0.48</td>
</tr>
<tr>
<td>Wolof</td>
<td>0.51 ± 0.32</td>
<td>0.47 ± 0.32</td>
<td>0.41 ± 0.02</td>
<td>0.26 ± 0.39</td>
<td>0.25 ± 0.39</td>
<td>0.3 ± 0.43</td>
<td>0.27 ± 0.39</td>
</tr>
<tr>
<td>Yoruba</td>
<td>0.48 ± 0.07</td>
<td>0.36 ± 0.07</td>
<td>0.43 ± 0.06</td>
<td>0.33 ± 0.45</td>
<td>0.09 ± 0.05</td>
<td>0.16 ± 0.11</td>
<td>0.09 ± 0.05</td>
</tr>
</tbody>
</table>

Figure 12: Probing classifiers’ results by language and task category. White squares denote that the morphosyntactic category is not represented in the language.

4.9.2 Results

Probing Table 12 presents the results of probing experiments averaged over the probing tasks and experiment runs within each language. The overall pattern is that BLOOM-1B7 performs on par or better than BLOOM, and both LLMs outperform the count-based baselines. In particular, the LLMs achieve more robust performance on Arabic, Basque, and Indo-European languages (e.g., Catalan, French, Hindi, Portuguese, Spanish, and Urdu),

43. BertTokenizer: hf.co/bert-base-multilingual-cased
44. XLMRobertTokenizer: hf.co/xlm-roberta-base
Table 13: Results of statistical tests and correlation analysis between probing performance and linguistic, dataset, and model configuration criteria.

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Model</th>
<th>Test</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language script</td>
<td>BLOOM</td>
<td>Mann-Whitney U</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>BLOOM-1B7</td>
<td></td>
<td>0.13</td>
</tr>
<tr>
<td>Language family</td>
<td>BLOOM</td>
<td>ANOVA</td>
<td>&lt;0.01</td>
</tr>
<tr>
<td></td>
<td>BLOOM-1B7</td>
<td></td>
<td>&lt;0.01</td>
</tr>
<tr>
<td>Probing dataset size</td>
<td>BLOOM</td>
<td>Pearson</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>BLOOM-1B7</td>
<td></td>
<td>0.02</td>
</tr>
<tr>
<td>Pretraining dataset size</td>
<td>BLOOM</td>
<td>Pearson</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>BLOOM-1B7</td>
<td></td>
<td>&lt;0.01</td>
</tr>
</tbody>
</table>

while Bengali, Wolof, and Yoruba receive the lowest scores. We attribute this behavior to the transfer abilities: BLOOM infers linguistic properties better for the closely related languages that comprise a significant amount of data. For example, the performance on any Romance language is better than in English, and the results in Indic languages are close to those in high-resource languages.

Figure 12 presents the language-wise probing performance results for morphosyntactic features represented at least in 5 languages. The probing performance of both LLMs is similar despite the difference in size. We find that the LLMs infer Mood and Person well with no regard for language. Number, NumType (numeral type), and Voice are moderately inferred in most languages. The models generally show worse qualities in the other categories, indicating that they do not encode such morphological information. The possible explanation of such difference in performance may be the diversity of possible values of these categories. For example, Mood and Person share similar values across the presented languages, while the set of Case values is highly dependent on the language.

Correlation The correlation analysis results support conclusions on the probing performance and reveals contributing factors (see Table 13). Both models show similar results on the languages with different language scripts. Results of BLOOM-1B7 are highly correlated with language family, probing dataset size, and pretraining dataset size. According to the results of Mann-Whitney U test, BLOOM-1B7 shows significantly better results ($p < 0.01$) than BLOOM. However, BLOOM shows more stable performance on different languages in spite of the amount of data it has seen during pretraining. This might indicate the better generalization abilities of the model with more parameters.

Discussion It should be noted that the following questions remain for further research:

1. Generalizing abilities. BLOOM-1B7 is leading in the average performance of morphosyntactic feature classification for the languages in Table 12. The BLOOM results are lower, which can be interpreted as a worse grammatical generalization over the aforementioned languages. However, the BLOOM-1B7’s probing correlation results with
factors like pretraining dataset size are more prominent, which makes it potentially less generalizing on the under-resourced languages than the bigger version.

2. Multilingual abilities. A separate research interest implies considering languages that are not explicitly included in the pretraining corpus of the models. Expanding the set of languages for probing will allow for a typological interpretation and a deeper analysis of the most learnable and hard-to-learn linguistic features on a more considerable scope.

3. Under-resourced language evaluation. The under-resourced languages of the Indic and Niger-Congo families included in the pretraining corpus in smaller shares represent a separate subject for future probing. We also plan to investigate the results of high-resourced and under-resourced languages to reveal possible linguistic insights in these two groups.

4. Different layers and training dynamics. The analysis has focused on averaged representations of all layers and at the end of training. Analyzing different layers may reveal how morpho-syntactic representations are built during processing. Similarly, investigating how properties are acquired over the course of pre-training (Choshen et al., 2022; Zhang et al., 2021; Voloshina et al., 2022) is a viable direction for research.

4.10 Bias

As a preliminary study into the biases learned by BLOOM, we present evaluation on the multilingual CrowS-Pairs dataset, which combines a revised version of the CrowS-Pairs dataset developed by Nangia et al. (2020) together with the French version of CrowS-Pairs introduced by Névéol et al. (2022). One challenge of this evaluation was to adapt a dataset originally intended for masked language models to autoregressive language models such as BLOOM. CrowS-Pairs relies on minimal pairs to compare a stereotyped statement and a non-stereotyped statement (e.g. “Women can’t drive.” is a gender stereotype while “Men can’t drive” is not). The two statements differ only by the social category targeted by the stereotype and that social category is present in the stereotyped statement and not in the non-stereotyped statement. The evaluation aims at assessing systematic preference of models for stereotyped statements. The original “metric score” compared pseudo-log-likelihood of sentences in a pair to determine which sentence received a higher score from a masked language model. Prompts were designed to require the model to select one of the statements based on the “likely” and “realistic” nature of the situations described.

Figure 13 shows that BLOOM’s overall prompt accuracy was close to .50, which suggests an overall absence of bias. We note that the scores in English and French are very close, suggesting similar overall behavior of the model on both languages. We also show results on mono-lingual autoregressive models — GPT-Neo (Black et al.) and GPT-FR (Simoulin and Crabbé, 2021) for English and French, respectively.

Table 14 presents the results per bias type in the CrowS-Pairs dataset. The results are quite homogeneous over the categories, which contrasts with previous studies on masked language models, which suggested models were prone to bias in specific categories, which differed between models tested. Nonetheless, accuracy significantly differs from 50 (T-test,
Figure 13: Overall accuracy of BLOOM on crowS-Pairs per prompt for English and French. Results on the two smallest BLOOM models and monolingual GPT models of comparable size are also shown.

<table>
<thead>
<tr>
<th>Bias type</th>
<th>support</th>
<th>English</th>
<th>French</th>
</tr>
</thead>
<tbody>
<tr>
<td>ethnicity, color</td>
<td>460</td>
<td>50.05</td>
<td>50.48*</td>
</tr>
<tr>
<td>gender</td>
<td>321</td>
<td>51.17*</td>
<td>51.24*</td>
</tr>
<tr>
<td>socioeconomic status</td>
<td>196</td>
<td>51.05*</td>
<td>52.22*</td>
</tr>
<tr>
<td>nationality</td>
<td>253</td>
<td>49.25*</td>
<td>48.49*</td>
</tr>
<tr>
<td>religion</td>
<td>115</td>
<td>53.82*</td>
<td>53.01*</td>
</tr>
<tr>
<td>age</td>
<td>90</td>
<td>49.35</td>
<td>50.13</td>
</tr>
<tr>
<td>sexual orientation</td>
<td>91</td>
<td>50.00</td>
<td>49.9</td>
</tr>
<tr>
<td>physical appearance</td>
<td>72</td>
<td>48.20</td>
<td>49.67</td>
</tr>
<tr>
<td>disability</td>
<td>66</td>
<td>48.49*</td>
<td>49.16*</td>
</tr>
<tr>
<td>other</td>
<td>13</td>
<td>50.18</td>
<td>42.1*</td>
</tr>
<tr>
<td>All</td>
<td>1,677</td>
<td>49.78*</td>
<td>50.61*</td>
</tr>
</tbody>
</table>

Table 14: BLOOM accuracy results on crowS-Pairs bias categories averaged over eight runs for English and French. Significance for the one sample T-test ($p < .05$) is indicated with *.

$p < .05$ overall for both languages, as well as for a number of bias categories, as shown per asterisks in the table.

**Limitations** Blodgett et al. (2021) discuss validity issues with the original CrowS-Pairs corpus. The CrowS-Pairs version used here differs from the original by addressing some of the issues pointed out by Blodgett et al. (2021) and by constructing 200 additional sentence pairs based on stereotypes collected from French speakers. In a recent evaluation of bias in masked language models in English and French, results obtained on the revised dataset were not significantly different from those obtained on the original dataset Névéol et al. (2022).
However, its original validation does not naturally apply here, and comparison to other CrowS-Pairs results is more difficult. For a stronger assessment of bias, results obtained with CrowS-Pairs should be compared with other measures of bias, and also assessed for all languages in the model. However, as noted by Talat et al. (2022), very little material (corpora, measures) is available for multilingual bias assessment.

Although our examinations suggest a limited presence of bias in the model, they cannot cover the breadth of possible usage scenarios. One such scenario where models may have a larger impact is on linguistic diversity and language variation encountered. As the training resources for BLOOM are carefully curated, they may also capture some language variations to a larger degree than other models. This also impacts the ability of trained models to equitably represent different variations. Such differences can aid in the propagation and legitimization of some language variants over others. Our evaluation of biases in the model are further limited to the situations, languages and language variants that are covered by multilingual CrowS-Pairs. We therefore expect a distinction between our findings using CrowS-Pairs and wider model use (for a more detailed exploration on such differences, see Raji et al., 2021).

5. Conclusion

In this work, we present BLOOM, a 176B-parameter open-access multilingual language model. BLOOM was created by BigScience, a collaboration of hundreds of researchers, and was trained on the French government-funded Jean Zay supercomputer for 3.5 months. In this paper, we chronicled the development of BLOOM, from the creation of its training dataset ROOTS to the design of its architecture and tokenizer. We also discuss evaluation results of BLOOM and other large language models, finding it has competitive performance that improves after multitask finetuning.

We hope that the release of a powerful multilingual language model unlocks new applications and research directions for large language models. Further, we hope that documenting our experience will help the machine learning research community organize new large-scale collaborative projects similar to BigScience. Besides enabling results that are impossible for any individual research group to achieve, this form of organization will also allow more people with different backgrounds to share their ideas and participate in the development of major advances in the field.

6. Contributions

Authors are assigned to each authorship category according to which aspects of the project they contributed to. Many authors appear under multiple categories because they contributed to the project in more than one way. Author order in all categories is alphabetical by first name, except for “Major Contributors” where authors are shuffled randomly apart from Teven Le Scao, who is intentionally listed first and “Organization” where Thomas Wolf is intentionally listed last. A description of each category follows. For finer-grained contribution details, please see the papers mentioned under each category.

Major Contributors lists individuals without whom BLOOM would not have happened and/or who spent more than 20% of their time on the BigScience effort as a whole.
**Dataset** lists individuals who contributed to data sourcing, organization, and processing efforts, including the authors of Laurençon et al. (2022), McMillan-Major et al. (2022), and Jernite et al. (2022).

**Tokenization** lists individuals who built the BLOOM tokenizer and authors of Mielke et al. (2021).

**Prompt Engineering** lists individuals who wrote, edited, and reviewed prompt templates for the datasets we consider as well as authors of Sanh et al. (2022), Bach et al. (2022), and Muennighoff et al. (2022b).

**Architecture and Objective** lists individuals who ran experiments to help determine BLOOM’s model architecture and training objective, including authors of Wang et al. (2022a) and Le Scao et al. (2022).

**Engineering** lists individuals who contributed to code and infrastructure to train BLOOM on the Jean Zay supercomputer.

**Evaluation and interpretability** lists individuals who helped evaluate the BLOOM model as well as authors of Talat et al. (2022).

**Broader Impacts** lists authors of the ethical charter, license, and model card, in addition to individuals who studied privacy issues, social impacts, and BLOOM’s carbon footprint.

**Applications** lists members of working groups focused on applications of BLOOM, including authors of Fries et al. (2022b), Fries et al. (2022a), and De Toni et al. (2022).

**Organization** lists individuals who coordinated the BigScience effort and authors of Akiki et al. (2022).

**Acknowledgments**

The BigScience Workshop was granted access to the HPC resources of the 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 the Grand équipement national de calcul intensif (GENCI). Model training ran on the Jean-Zay supercomputer of GENCI at IDRIS, and we thank the IDRIS team for their responsive support throughout the project, in particular Rémi Lacroix.

Roman Castagné, Thomas Wang, Benoît Sagot and Rachel Bawden’s contributions were funded by Benoît Sagot’s and Rachel Bawden’s chairs in the PRAIRIE institute funded by the French national agency ANR as part of the “Investissements d’avenir” programme under the reference ANR-19-P3IA-0001. Aurélie Névéol’s contribution was supported by ANR under grant GEM ANR-19-CE38-0012. Oskar van der Wal’s contributions were financed by the Dutch Research Council (NWO) as part of Open Competition Digitalisation-SSH with project number 406.DI.19.059.
The BigScience Workshop would also like to acknowledge the support and financing of the following organizations, organization members and affiliations of some of the participants: ESPCI and LAMSADE (Dauphine Université, PSL, CNRS) for Alexandre Allauzen; MELODI team at IRIT/University of Toulouse for Farah Benamara, Chloé Braud, Philippe Muller, and Véronique Moriceau; IRISA LinkMedia team IMATAG/CNRS for Vincent Claveau and Antoine Chaffin; Université de Lorraine ATILF UMR 7118 CNRS / UL for Mathieu Constant; University of Paris for Benoît Crabbé, Marie Candito and Antoine Simoulin; GdR TAL (CNRS) for Béatrice Daille; CNRS DR1 INSERM UMR1093 UBFC Dijon for Peter Ford Dominey; Aix-Marseille University UTLP CNRS LIS/UMR7220 for Benoît Favre and Frédéric Béchet; CEA LASTI for Bertrand Delezoïde, Olivier Ferret, Adrian Popescu and Julien Tourille; Sorbonne Université LORIA for Karen Fort; CNRS DR1 LORIA UMR7503 Nancy for Claire Gardent and Christophe Cerisara; MAS Laboratory of Ecole Centrale Paris for Céline Hudelot, RCLN/LIPN UMR 7030 University Sorbonne-Paris-Nord/CNRS for Joseph Le Roux and Nadi Tomeh, Université de Paris and Necker - Enfants Malades hospital for Antoine Neuraz and Ivan Lerner, Université Paris Saclay LISN CNRS UMR9105 for Aurélie Névéol, Anne-Laure Ligozat, Caio Corro, François Yvon; Inria, Univ. Bordeaux and Ensta ParisTech for Pierre-Yves Oudeyer, Cédric Colas, Grégol Kovac, Tristan Karch; Inria Paris for Benoît Sagot, Djamé Seddah, Pedro Ortiz; University Toulouse CNRS for Ludovic Tanguy, Sorbonne Université, LIMICS (Sorbonne Université, Inserm, Univ. Sorbonne Paris Nord) for Xavier Tannier; I3S Laboratory, CNRS, INRIA, Université Côte d’Azur for Serena Villata and Elena Cabrio; Airbus, Central Research & Technology for Guillaume Alleon, Alexandre Arnold, and Catherine Kobus; Cloud Temple for Jean-Michel Dussoux; Illuin Technology for Robert Vesoul, Gautier Vialaud, Martin d’Hoffschmidt, and Wacim Belblidia; Levia.ai for Romain Riviere; LightOn for Igor Carron, Laurent Daudet, Iacopo Poli, and Julien Launay; Nabla for Alexandre Lebrun, Martin Raison, and Samuel Humeau; Naver Labs Europe for Matthias Gallé and Laurent Besacier; Orange Labs for Géraldine Dammati, Johannes Heincke, and Frederic Herledan; OVHcloud for Jean-Louis Queguiner and Guillaume Salou; ReciTAL for Thomas Scialom, Gilles Moysse, and Jacopo Staiano; Renault Group for Vincent Feuillard, Joan André, Francois-Paul Servant, Raphael Sourt, and Ayhan Uyanik; SYSTRAN for Jean Senellart, Josep Crego, Elise Michon, Guillaume Klein, Dakun Zhang, and Natalia Segal; Ubisoft for Guillaume Gaudron. Leipzig University and the Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) in Leipzig for Christopher Akiki.

Hugging Face provided storage for the entirety of the project, as well as compute for development and part of training the smaller BLOOM models. Many of the evaluations in this paper were made possible by compute resources donated by CoreWeave and EleutherAI.

References


Anthony Moi, Pierric Cistac, Nicolas Patry, Evan P. Walsh, Funtowicz Morgan, Sebastian Pütz, Thomas Wolf, Sylvain Gugger, Clément Delangue, Julien Chaumond, Lysandre


Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery


Appendix A. Prompts

The following contains prompts used for evaluation. The prompts are also available in PromptSource (Bach et al., 2022). A sample with a prompt applied as well as the raw prompts are provided. For raw prompts, double curly brackets are filled with content from the sample when used.

Contents

A.1 SuperGLUE/wsc.fixed ................................................. 66
  A.1.1 Data example .................................................. 66
  A.1.2 Prompts ...................................................... 66
A.2 SuperGLUE/wic ......................................................... 66
  A.2.1 Data example .................................................. 66
  A.2.2 Prompts ...................................................... 67
A.3 SuperGLUE/boolq ......................................................... 67
  A.3.1 Data example .................................................. 67
  A.3.2 Prompts ...................................................... 67
A.4 SuperGLUE/axb & SuperGLUE/axg ................................ 68
  A.4.1 Data example .................................................. 68
  A.4.2 Prompts ...................................................... 68
A.5 XNLI & SuperGLUE/CB .............................................. 68
  A.5.1 Data example .................................................. 68
  A.5.2 Prompts ...................................................... 69
A.6 XWinograd ............................................................... 69
  A.6.1 Data example .................................................. 69
  A.6.2 Prompts ...................................................... 69
A.7 XCOPA & SuperGLUE/COPA ......................................... 70
  A.7.1 Data example .................................................. 70
  A.7.2 Prompts ...................................................... 70
A.8 XStoryCloze & Story Cloze ......................................... 71
  A.8.1 Data example .................................................. 71
  A.8.2 Prompts ...................................................... 71
A.9 WMT ................................................................. 71
  A.9.1 Data example .................................................. 71
  A.9.2 Prompts ...................................................... 72
A.10 DiaBLa .............................................................. 72
  A.10.1 Data example .................................................. 72
  A.10.2 Prompt ....................................................... 72
A.11 Flores-101 (MT) ..................................................... 72
  A.11.1 Data example .................................................. 73
  A.11.2 Prompt ....................................................... 73
A.12 CrowS-Pairs .......................................................... 73
  A.12.1 Data example .................................................. 73
  A.12.2 Prompts ...................................................... 73
A.1 SuperGLUE/wsc.fixed

A.1.1 Data example
Prompt name: GPT-3 Style

Passage: I tried to paint a picture of an orchard, with lemons in the lemon trees, but they came out looking more like light bulbs.

Question: In the passage above, does the pronoun "they" refer to lemon trees?

Answer: No

A.1.2 Prompts
GPT-3 Style

Passage: {{ text }}

Question: In the passage above, does the pronoun "{{ span2_text }}" refer to {{ span1_text }}?

Answer: replaced with

{{ text }} In the previous sentence, can the pronoun "{{ span2_text }}" be replaced with "{{ span1_text }}"? Yes or no?

the pronoun refers to

{{ text }} In the passage above, the pronoun "{{ span2_text }}" refers to {{ span1_text }}. True or false?

does p stand for

{{ text }} Here, does "{{ span2_text.lower() }}" stand for {{ span1_text }}? Yes or no?

the pronoun refers to

{{ text }} In the passage above, the pronoun "{{ span2_text }}" refers to {{ span1_text }}. True or false?

A.2 SuperGLUE/wic

A.2.1 Data example
Prompt name: GPT-3 Style

As he called the role he put a check mark by each student’s name. A check on its dependability under stress.

Question: Is the word 'check' used in the same sense in the two sentences above?
A.2.2 Prompts

GPT-3 Style

{{sentence1}}

{{sentence2}}

Question: Is the word "{{word}}" used in the same sense in the two sentences above?

question-context-meaning-with-label

Does the word "{{word}}" have the same meaning in these two sentences? Yes, No?

GPT-3 prompt-with-label

{{sentence1}}

{{sentence2}}

Question: Is the word "{{word}}" used in the same sense in the two sentences above? Yes, No?

polysemous

The word "{{word}}" has multiple meanings. Does it have the same meaning in sentences 1 and 2? Yes or no? Sentence 1: {{sentence1}} Sentence 2: {{sentence2}}

similar-sense

{{sentence1}}

{{sentence2}}

Similar sense of {{word}}?

A.3 SuperGLUE/boolq

A.3.1 Data example

Prompt name: GPT-3 Style

Phantom pain -- Phantom pain sensations are described as perceptions that an individual experiences relating to a limb or an organ that is not physically part of the body. Limb loss is a result of either removal by amputation or congenital limb deficiency. However, phantom limb sensations can also occur following nerve avulsion or spinal cord injury.

Question: is pain experienced in a missing body part or paralyzed area?

Answer: Yes

A.3.2 Prompts

GPT-3 Style

{{ passage }}

Question: {{ question }}

Answer:

yes_no_question

Text: {{passage}}

Answer the following yes/no question: {{question}}? Yes or no?
A.4 SuperGLUE/axb & SuperGLUE/axg

A.4.1 Data example

Prompt name: GPT-3 style

The taxpayer met with the accountant to get help filing his taxes.

Question: The accountant sought help filing taxes. True or False?

Answer: False

A.4.2 Prompts

GPT-3 style

Using only the above description and what you know about the world, is "{{sentence2}}" definitely correct? Yes or no?

can we infer

Suppose {{sentence1}} Can we infer that "{{sentence2}}"? Yes or no?

guaranteed true

Given {{sentence1}} Is it guaranteed true that "{{sentence2}}"? Yes or no?

justified in saying

{{sentence1}} Are we justified in saying that "{{sentence2}}"? Yes or no?

A.5 XNLI & SuperGLUE/CB

A.5.1 Data example

Prompt name: GPT-3 style

Well, I wasn’t even thinking about that, but I was so frustrated, and, I ended up talking to him again.

Question: I havent spoken to him again. True, False, or Neither?

Answer: False
A.5.2 Prompts

GPT-3 style

{{premise}} \n\nQuestion: {{hypothesis}} True, False, or Neither?

MNLI crowdsourcing

{{premise}} Using only the above description and what you know about the world, "{{hypothesis}}" is definitely correct, incorrect, or inconclusive?

can we infer

Suppose {{premise}} Can we infer that "{{hypothesis}}"? Yes, no, or maybe?

guaranteed/possible/impossible

Assume it is true that {{premise}} \n\nTherefore, "{{hypothesis}}" is {{"guaranteed"}}, {{"possible"}}, or {{"impossible"}}?

justified in saying

{{premise}} Are we justified in saying that "{{hypothesis}}"? Yes, no, or maybe?

A.6 XWinograd

A.6.1 Data Example

Prompt name: Replace

The city councilmen refused the demonstrators a permit because _ feared violence. \nReplace the _ in the above sentence with the correct option: \n- the demonstrators \n- The city councilmen

Answer: The city councilmen

A.6.2 Prompts

Replace

{{sentence}} \n\nReplace the _ in the above sentence with the correct option:
- {{option1}}
- {{option2}}

True or False

The _ in the sentence below refers to {{option1}}. True or False? {{sentence}}

does underscore refer to

{{sentence}} In the previous sentence, does _ refer to {{ option1 }} or {{ option2 }}?
underscore refer to

What does the _ in the above sentence refer to?
{{ option1 }} or {{ option2 }}?

stand for

In the sentence below, does the _ stand for {{answer_choices[0]}} or {{answer_choices[1]}}?

A.7 XCOPA & SuperGLUE/COPA

A.7.1 DATA EXAMPLE

Prompt name: C1 or C2? premise, so/because...

"It was fragile." or "It was small."? The item was packaged in bubble wrap. because

Answer: It was fragile.

A.7.2 PROMPTS

C1 or C2? premise, so/because...

{{ answer_choices[0] }}" or "{{ answer_choices[1] }}"? {{ premise }}
{% if question == "cause" %} because {% else %} so {% endif %}

best_option

{{ premise }}

What's the best option?

We are looking for {% if question == "cause" %} a cause {% else %} an effect {% endif %}

cause_effect

{{ premise }}

Select the most plausible {% if question == "cause" %} cause: {% else %} effect: {% endif %}

i_am_hesitating

{{ premise }}

I am hesitating between two options. Help me choose the more likely {% if question == "cause" %} cause: {% else %} effect: {% endif %}

plausible_alternatives

{{ premise }}

This happened because... As a consequence... Help me pick the more plausible option:
A.8 XStoryCloze & Story Cloze

A.8.1 Data example

XStoryCloze and Story Cloze are not publicly available datasets. Please contact the authors of Lin et al. (2021) for XStoryCloze and Mostafazadeh et al. (2017) for Story Cloze samples.

A.8.2 Prompts

Answer Given options

{{input_sentence_1}} {{input_sentence_2}} {{input_sentence_3}}
{{input_sentence_4}} What is a possible continuation for the story given the following options? - {{answer_choices | join("\n- ")}}

Choose Story Ending

Read the following story:

{{input_sentence_1}}
{{input_sentence_2}}
{{input_sentence_3}}
{{input_sentence_4}}
Choose a possible ending for the previous story from the following options: \n- {{answer_choices | join("\n- ")}}

Story Continuation and Options

What is a possible continuation for the following story? \n
{{input_sentence_1}}
{{input_sentence_2}}
{{input_sentence_3}}
{{input_sentence_4}}
Choose from the following options: \n- {{answer_choices | join("\n- ")}}

Generate Ending

Generate a possible ending for the following story: {{input_sentence_1}}
{{input_sentence_2}}
{{input_sentence_3}}
{{input_sentence_4}}

Novel Correct Ending

I read the following novel: {{input_sentence_1}}
{{input_sentence_2}}
{{input_sentence_3}}
{{input_sentence_4}}
What do you think is the most probable ending? You can choose from the following options: - {{answer_choices | join("\n- ")}}

A.9 WMT

Prompts for Section 4.3.1, where we compare prompts in both zero-shot and 1-shot settings for four language directions (en↔{hi,fr}).

A.9.1 Data example

The prompt names and content are specific to the language direction. The prompts below each exist in four versions, where "l1" and "l2" are replaced by the language codes of the source and target languages respectively (en, fr or hi) and “L1” and “L2” are replaced by the language names of the source and target languages respectively (English, French or Hindi).

Prompt name: a_good_translation-l1-l2-source+target

Given the following source text in English: Spectacular Wingsuit Jump Over Bogota, a good French translation is:

Answer: Spectaculaire saut en "wingsuit" au-dessus de Bogota

71
A.9.2 Prompts

a_good_translation-l1-l2-source+target

Given the following source text in L1: "{{translation[l1]}}", a good L2 translation is: "||| {{translation[l2]}}"

gpt-3-l1-l2-target

Q: What is the "{{L2}}" translation of "{{translation[l2]}}"? A:

version-l1-l2-target

If the original version says: "{{translation[l1]}}"; then the L2 version should say:

xglm-l1-l2-source+target

{{L1}}: "{{translation[l1]}}" = {{L2}}: "||| {{ref}}"

A.10 DiaBLa

Prompts for contextual MT results shown in Table 7.

A.10.1 Data example

Prompt name: xglm-source+target

English: We appear to have stopped moving. = French:

Answer: "J'ai l'impression qu'on s'est arrêtés."

A.10.2 Prompt

xglm-source+target

{% set trg_lang ="French" %}{% set src_lang ="English" %}
{% if utterance_meta.lang == "french" %}{% set trg_lang = "English" %}{% set src_lang = "French" %}{% endif %}

{{ src_lang }}: {{ orig }} = {{ trg_lang }}: "||| {{ ref }}"

The dialogue set is bilingual (between native English and native French speakers). In few-shot setups, few-shot examples are by default in either language direction (English→French or French→English) regardless of the direction of the current example. We implemented some additional DiaBLa tasks that control the language direction of the few-shot example to be the same or the opposite direction to the current example (diabla_1_shot_context_same and diabla_1_shot_context_opposite).

A.11 Flores-101 (MT)

The prompts are specific to each language pair as indicated by the source and target languages in the prompt name. Below is an example for French→Catalan translation.
A.11.1 Data example

Prompt name: xglm-French-Catalan-source+target

French: L’aménagement vieillot, l’absence d’équipements récents et un certain charme à l’ancienne font également partie de leur caractère. = Catalan: |||
Els accessoris vintage, la manca de les comoditats modernes i un cert envelliment elegant també formen part del seu caràcter

A.11.2 Prompt

xglm-French-Catalan-source+target

French: {{ sentence_fra }} = Catalan: ||| {{ sentence_cat }}

A.12 CrowS-Pairs

A.12.1 Data example

Sentence A: He couldn’t figure out the issue with the rope, but he tried not to get too down on himself, most black kids don’t have the best education growing up.

Sentence B: He couldn’t figure out the issue with the rope, but he tried not to get too down on himself, most white kids don’t have the best education growing up.

Select the option that is the most plausible:
- Sentence A demonstrates a stereotype
- Sentence B violates a stereotype

Answer: Sentence A demonstrates a stereotype

A.12.2 Prompts
demonstrate_or_violate

Sentence A: {{sent_more}}
Sentence B: {{sent_less}}
Select the option that is the most plausible:
- {{answer_choices[0]}}
- {{answer_choices[1]}}