Université de Lorraine

ED77 - IAEM Lorraine

UMR 7503 - Laboratoire lorrain de recherche en informatique et ses applications

Scaling multilingual language models under constrained data

Thèse

Présentée et soutenue publiquement le 18 décembre pour l'obtention du doctorat de l'Université de Lorraine (mention informatique) par Teven Le Scao

Abstract / Résumé

Abstract

This work is concerned with large language models (LLMs), the current dominant paradigm in natural language processing in research and industrial settings, thanks to those models' generalization capabilities. Although LLMs have generated wide excitement, many obstacles are left before they can be deployed everywhere, not the least of which is the fact that state-of-the-art results are, for now, only possible in English, with other languages far behind.

We try to make progress towards better models in languages with lower amounts of linguistic resources. In the first half, we create artifacts: we process and crowdsource a large multilingual dataset for language modeling and document the tools required for this, and then we train the first open massive multilingual language model as part of a large international research collaboration. In the latter half, we propose empirically derived recommendations for practitioners in low-resource pretraining and finetuning settings. We estimate the performance degradation that results from training on multiple epochs in compute-rich, data-constrained settings and find that it is broadly acceptable. We then show that the prompt-following capabilities of language models can be combined with the pretrain-then-finetune paradigm in NLP and that this helps particularly in settings where downstream task data is scarce.

Résumé

Cette thèse s'intéresse aux grands modèles de langage (LLMs), le paradigme actuellement dominant de traitement du langage naturel dans la recherche et l'industrie grâce aux capacités de généralisation de ces modèles. Bien que les LLM aient suscité un vif engouement, de nombreux obstacles demeurent avant qu'ils ne puissent être déployés partout, un des principaux étant que les résultats de pointe ne sont, pour l'instant, possibles qu'en anglais, les autres langues étant bien en retard.

Nous essayons dans ce travail de progresser vers de meilleurs modèles dans les langues disposant de moins de ressources d'entraînement. Dans une première partie, nous créons des artefacts : nous traitons et mettons en commun une grande base de données multilingue pour la modélisation du langage et documentons les outils nécessaires pour cela, puis nous entraînons le premier modèle de langage massivement multilingue dans le cadre d'une grande collaboration internationale de recherche. Dans une deuxième partie, nous proposons des recommandations empiriquement dérivées pour l'entraînement et le peaufinage sur des ressources de faible quantité. Nous estimons la dégradation des performances qui résulte de l'entraînement sur plusieurs époques dans des environnements riches en puissance de calcul mais pauvres en données et constatons qu'elle est largement acceptable. Nous montrons ensuite que les capacités des modèles de langage à suivre des instructions peuvent être combinées avec le paradigme d'entraînement puis peaufinage, et que cela aide particulièrement dans les situations où les données pour les tâches en aval sont rares.

Résumé long

En l'espace de quelques années seulement, les modèles de langage, qui consistent en des estimateurs de la probabilité d'une séquence de texte, sont passés d'un exercice principalement académique - faire réimplémenter le lissage Kneser-Ney aux étudiants de première année, puis passer aux problèmes intéressants - à une véritable industrie. En ce court laps de temps, ces techniques se sont avérées être un outil de travail fiable pour un large éventail de problèmes en traitement du langage naturel (TAL, NLP en anglais), se hissant d'abord au sommet des benchmarks de recherche, puis dépassant totalement ces derniers, grâce à leurs propriétés favorables de dimensionnement et à l'augmentation constante des données et des capacités de calcul qui permettent de nourrir cette mise à l'échelle. Une thèse maximaliste pourrait avancer que tous les problèmes peuvent être exprimés sous forme de texte, et que les grands modèles de langage (LLMs) sont actuellement en train de résoudre le traitement de texte en général; ils devraient même bientôt se déployer dans d'autres modalités, associés à des encodeurs d'images et d'audio et intégrés dans des algorithmes d'apprentissage par renforcement.

Malgré l'enthousiasme qu'ils suscitent, cependant, les LLMs ont encore un long chemin à parcourir. Un débat animé est encore en cours sur l'étendue de leurs capacités, et en particulier sur leur aptitude à généraliser. Le manque de transparence croissant dans le domaine n'aide pas : avec seulement quelques détails sur les modèles les plus puissants, et aucune autre façon de les interroger qu'à travers des API en boîte noire, il est difficile d'examiner leurs compétences. Quelle que soit l'issue de ce débat, il existe cependant un cadre dans lequel nous savons qu'ils sont encore très imparfaits : dans les environnements non anglophones et plus généralement multilingues, les meilleurs modèles de langage ne sont pas à la hauteur des promesses de la thèse maximaliste, principalement freinés par un manque de données d'entraînement.

Notre objectif à long terme est de rendre cette technologie plus largement disponible dans toutes les langues. Concrètement, pour le moment, cela signifie qu'il faut rendre possible l'entraînement de modèles de langage dans les langues à faibles ressources, ce qui est ici une dénomination très large - les ensembles de données de milliers de milliards de mots qui ont été utilisés pour entraîner la dernière génération de LLM ne sont à la disposition que de très peu de communautés linguistiques. Cette thèse comporte trois contributions à cet objectif : dans le cadre d'une grande collaboration internationale de recherche, nous compilons, traitons et documentons un grand ensemble de données multilingues pour la modélisation du langage, ainsi que les outils que nous avons utilisés pour le créer ; nous utilisons ensuite cet ensemble de données pour former le premier grand modèle de langage massivement multilingue, dans le but de diffuser tout le savoir-faire nécessaire pour des projets de cette envergure ; et enfin, dans un travail plus théorique inspirés par nos difficultés dans la première partie, nous déduisons de manière empirique un ensemble de recommandations actionnables pour les chercheurs dans les environnements à données limitées.

Dans le **premier chapitre**, une présentation générale, nous commençons par définir quelques termes: les différents types de modèles, des premières expériences humaines aux modèles modernes basés sur les *transformers*, puis les différents objectifs de modélisation de langage, en se concentrant sur les modèles causaux (ou autoregressifs) et les modèles à masques. Nous présentons ensuite une brève histoire de l'évolution des modèles de langage, de composants pour d'autres tâches telles que la reconnaissance vocale et la traduction automatique aux modèles pré-entraînés modernes. La puissance de calcul étant devenue plus disponible avec le temps, les méthodes universelles qui utilisent le calcul de manière efficace ont finalement été favorisées, avec l'émergence d'un nouveau champ de recherche visant à obtenir des capacités de multi-tâches à partir des modèles, en donnant des instructions en langage naturel pour expliquer chacune des tâches. Finalement, nous montrons comment cela a permis l'apparition du zero-shot, ou classification sans données spécifiques, qui est devenue la principale métrique pour évaluer les modèles de langage de dernière génération, en particulier sur des tâches de génération ouvertes.

Une fois ces présentations faites, nous présentons comment les modèles de langage sont typiquement évalués. La dominance de BERT, puis la course entre BERT et ses successeurs, a été mesurée sur des benchmarks multi-tâches petite taille, contenant généralement une petite douzaine de tâches de NLP académiques et leurs données assorties. Ces benchmarks ont fini par être saturés par les modèles; cependant, avec l'essor de l'évaluation zero-shot comme méthode d'évaluation possible pour les modèles de langage, une grande partie de la communauté l'a adoptée comme moyen de continuer à progresser. Les modèles qui peuvent suivre des instructions sont maintenant évalués sur des des tâches comme MT-Bench et AlpacaEval, qui ont permis de contourner la difficulté d'obtenir des notes humaines pour les générations ouvertes en utilisant les modèles de pointe eux-mêmes. Finalement, nous concluons ce chapitre de présentation en examinant les attentes actuelles des LLMs. D'abord, l'utilisation de différentes modalités autres que le texte naturel. Un riche corpus de travaux dote les LMs de capacités dans d'autres modalités en les associant à des capteurs ou des actionneurs distincts dans de nouveaux domaines, tels que la vision, l'audio, ou les jeux. Mais surtout, la généralisation sur un corpus de données grand et varié qui comprend tous les problèmes que l'on voudrait leur poser. Cette section se termine sur la constatation que ces besoins sont principalement satisfaits en anglais, et définit l'objectif de cette thèse: faire des progrès dans les langues disposant de moins de ressources.

À mesure que les modèles de langage deviennent de plus en plus grands, la nécessité de disposer de vastes ensembles de données textuelles de haute qualité augmente, en particulier dans des contextes multilingues. Pour former des modèles de langage, il est nécessaire de disposer au préalable de données linguistiques pour les alimenter. Un **deuxième chapitre** documente donc les efforts de création et de curation de données entrepris au sein de Big-science, une collaboration internationale de recherche, pour assembler le corpus ROOTS, un ensemble de données multilingue de 1,6 téraoctets qui a été utilisé pour former la famille de modèles BLOOM, l'objet du chapitre d'après. Le corpus, ainsi que les outils utilisés pour le créer, ont tous été rendus disponibles au public.

Nous décrivons notre approche pour la compilation d'un jeu de données à l'échelle du web couvrant 59 langues, dont 46 langues naturelles et 13 langages de programmation. Étant donné l'importance que nous accordons à l'expertise des locuteurs natifs, le choix des langues a été principalement influencé par les communautés qui ont participé à l'effort. Notre corpus final se compose de deux composantes principales : 62% du texte provient d'une liste de sources de données linguistiques sélectionnées par la communauté et documentées, et 38% sont extraits d'un crawl web prétraité, OSCAR, filtré avec l'aide de locuteurs natifs. Notre objectif en construisant ROOTS était de soutenir les projets de modélisation monolingue et multilingue à grande échelle. Nous décrivons donc le processus de création et la composition finale de ROOTS, et les nombreux outils de données qui ont été développés tout au long du processus et qui nous ont permis de créer, d'obtenir, de nettoyer et d'inspecter l'ensemble des 498 jeux de données qui constituent ROOTS.

Nous avons présenté dans le premier chapitre un état du domaine excitant mais avec ses problèmes : les coûts de formation des grands modèles de langage ne sont possibles que pour les organisations bien dotées en ressources, et la plupart d'entre eux ne sont pas rendus publics. En conséquence, la majorité de la communauté de recherche n'est pas impliquée dans le développement et l'investigation des LLMs, et les grands modèles de langage sont presque toujours entraînés sur des textes en langue anglaise. Un **troisième chapitre** est consacré à un modèle de langage multilingue que nous appelons BLOOM (le BigScience Large Open-science Open-access Multilingual Language Model) qui a été développé et publié dans le cadre de BigScience, une collaboration de centaines de chercheurs, dans le but de résoudre ces problèmes.

À partir du jeu de données que nous avons construit dans le deuxième chapitre, nous avons pu entraîner un modèle de 176 milliards de paramètres sur 46 langues naturelles et 13 langages de programmation. Au moment de la rédaction, BLOOM était encore le plus grand modèle open-source jamais publié, le seul grand modèle de langage multilingue à grande échelle et le seul modèle de langage disponible au public de plus de 20 milliards de paramètres entraîné sur des données publiques. Tout d'abord, nous documentons l'intégralité de la conception de BLOOM, y compris les choix d'architecture et de modélisation, les décisions d'ingénierie et certaines variantes peaufinées, dans le but de démocratiser le savoir-faire autour de la formation de grands modèles de langage. Ensuite, nous rapportons les résultats de notre suite d'évaluation extensive, notamment la performance sur plusieurs tâches dans des contextes monolingues et multilingues, les effets du peaufinage, et une estimation du biais présenté par le modèle.

Le consensus sur les lois de dimensionnement des modèles de langage a évolué rapidement depuis les premières études à grande échelle. Avec une puissance de calcul fixe, nous savons maintenant que l'entraînement d'un modèle BLOOM de 57 milliards de paramètres sur 1,13 billion de tokens aurait donné de meilleures performances que celui que nous avons entraîné au chapitre précédent. Pour la plupart des langues, les données disponibles sont infiniment plus petites, ce qui signifie que les modèles de langage à grande échelle dans ces langues sont déjà limités par les données. Cela soulève la question du **quatrième chapitre** : que devrions-nous faire lorsque nous manquons de données ? Pour y répondre et aider les praticiens à décider comment allouer des ressources de calcul, nous examinons la dimensionnement des grands modèles de langage dans un régime contraint par les données. Afin de quantifier l'impact de la répétition des données, nous menons une série de plus de 400 expériences d'entraînement avec des contraintes variables en matière de données et de calcul, et enregistrons la perte de test finale. Nous utilisons ces résultats pour dériver une nouvelle loi de dimensionnement en données limitées qui généralise les travaux précédents.

Nous constatons que, bien que l'entraînement sans répétition donne systématiquement la meilleure perte de validation par puissance de calcul, ces différences sont insignifiantes parmi les modèles entraînés jusqu'à 4 époques et n'entraînent pas de différences dans les performances des tâches ultérieures. Des époques supplémentaires continuent d'être bénéfiques, mais les gains diminuent finalement jusqu'à zéro. Nous constatons également que, dans le régime contraint par les données, l'allocation optimale évolue légèrement en faveur de l'entraînement sur plus de données que ne le prévoient les lois existantes. Ces résultats suggèrent que nous pouvons continuer à augmenter les budgets de calcul d'entraînement plus loin dans l'avenir que prévu précédemment. Enfin, compte tenu des défis posés par les contraintes de données, nous envisageons des méthodes complémentaires à la répétition pour améliorer la précision des tâches ultérieures sans ajouter de nouvelles données en langue naturelle.

Après avoir consacré quelques chapitres à la création de modèles pré-entraînés, il est temps de les utiliser pour résoudre effectivement des tâches de traitement automatique du langage naturel. Dans le **cinquième chapitre**, nous comparons et unifions l'ancien paradigme, qui consistait à remplacer la tête de classification de tokens du modèle de base par une tête de classification explicite, puis à peaufiner le modèle, avec le nouveau paradigme, qui les utilise directement en tant que prédicteurs sans données à travers la génération de texte autorégressive, en décrivant la tâche avec une instruction. Les instructions peuvent aussi être utilisées dans la première méthode pour fournir des informations supplémentaires à la tâche du classificateur, en particulier dans le cas de faibles données. Il est donc naturel de se demander comment cela impacte l'efficacité en données du modèle, ou plus directement, combien de points de données vaut une instruction ?

Nous introduisons une métrique, l'avantage moyen des données, pour quantifier l'impact d'une instruction en pratique. Nous utilisons cette méthode pour unifier les deux paradigmes, en considérant la classification sans données comme un cas extrême du peaufinage qui utilise les instructions. Nous constatons que l'impact et la qualité des différentes instructions peut être quantifié en termes de données d'entraînement directes et qu'il varie en fonction de la nature des différentes tâches. Sur les tâches étudiées dans nos expériences, cet avantage est équivalent à 200 à 3500 points de données supplémentaires. Dans des les environnements de faibles à moyennes données, tels que les tâches rares ou les langues à ressources limitées, cet avantage peut constituer une véritable contribution à l'entraînement d'un modèle spécialisé.

Keywords *Transformers*

Table of contents

Chapter 1

Introduction

In the span of only a few years, language modeling, the problem of estimating the probability of text, has grown from a mostly academic exercise — make the first-year students re-implement Kneser-Ney smoothing, then move on to the interesting problems — to a full-fledged industry. During that time, language models have proved themselves to be a dependable workhorse across a wide range of problems in natural language processing (NLP), first claiming the top spot on research benchmarks, then moving beyond them entirely, all thanks to fortunate scaling properties, and to ever-growing amounts of data and computation to feed scaling. A maximalist thesis could go, all problems can be expressed in text, and large language models (LLMs) are currently solving text; soon they will even break out into other modalities, joined with image and audio encoders and integrated into reinforcement learning policies.

For all the excitement that they generate, however, LLMs still have a long way ahead of them. A fierce debate is still ongoing about the extent of their capabilities, and especially about their ability to generalize. The increasing lack of transparency in the field does not help: with only a few details about the strongest models, and no way to query them except through black-box APIs, it is difficult to investigate their abilities. There is a setting, however, in which we know they are still decidedly imperfect: in non-English and generally in multilingual settings, the best language models do not live up to the promise of the maximalist thesis, hampered mainly by a dearth of training data.

Our long-term goal is to make this technology broadly available in every language. Concretely, at the moment, this means making it possible to train language models in low-resource languages, which is here a broad umbrella - the datasets of trillions of words that were used to train the latest generation of LLMs are available to only very few linguistic communities. This thesis has three contributions towards that goal - as part of a large international research collaboration, we compile, process, and document a large massively multilingual dataset built for language modeling, as well as the tools we used to create it; we then use this dataset train the first massively multilingual large language model in the open, with the goal of spreading all the necessary know-how for projects of this scale; and finally, in follow-up work, we derive empirically a set of actionable recommendations for practitioners in data-constrained environments.

1.1 Thesis outline

Chapter 2 serves as an introduction to large language models. In it, we will introduce a bit of terminology, and then discuss the (recent) historical background that led to their development, and their rise to the dominant paradigm in NLP. We will finish with an overview of the capabilities expected from them and the ways researchers measure progress, with an eye toward current and future developments.

Chapters 3 and 4 together describe work performed as the training lead of BigScience, a 1-year international and multidisciplinary initiative formed with the goal of researching and training large language models in the open with the collaboration and scrutiny of as many members of the research community as possible. The main deliverables of BigScience, in addition to all of the research around large language models, were two artifacts. First, we released a massively multilingual corpus of text for language modeling, which we dubbed ROOTS, described in chapter 3, with a focus on proper documentation of the contained data and the rules and tools we used, so that future projects may learn from it. Second, a large language model trained on that corpus, which we dubbed BLOOM, described in chapter 4, with a thorough account of the training process and an extensive evaluation suite, again with the goal of democratizing language model training know-how.

Chapters 5 and 6 describe empirical work, inspired by the experience of creating the artifacts of the previous chapters, aimed at helping language modeling practitioners in data-constrained environments. In chapter 5, we investigate the scaling of large language models when compute is plentiful and training data is scarce, as is already the case for most languages and will be universal soon if computation trends hold. In it, we derive a generalization of scaling laws for the return on additional training on multiple epochs and the optimal allocation of resources in this case and investigate a few other ways of dealing with limited data. Finally, in chapter 6, we concern ourselves with adapting language models to downstream tasks. We bring together the two existing finetuning paradigms to measure the additional information contained within human prompts in terms of task-specific data points and show that prompt-based finetuning gets the best of both worlds. This technique is especially adapted to the many languages where strong pretrained models but not large amounts of task-specific data are available.

1.2 Publications

Parts of this thesis have appeared elsewhere in the following publications:

- T. Le Scao, T. Wang, D. Hesslow, L. Saulnier, S. Bekman, M. Saiful Bari, S. Biderman, H. Elsahar, J. Phang, O. Press, et al. What language model to train if you have one million GPU hours? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022b. URL https://aclanthology.org/2022.findings-emnlp.54/

- H. Laurençon, L. Saulnier, T. Wang, C. Akiki, A. Villanova del Moral, T. Le Scao, L. Von Werra, C. Mou, E. González Ponferrada, H. Nguyen, et al. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/ce9e92e3de2372a4b93353eb7f3dc0bd-Abstract-Datasets_and_Benchmarks.html

- T. Le Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022a. URL https://arxiv.org/abs/2211.05100
- N. Muennighoff, A. M. Rush, B. Barak, T. Le Scao, A. Piktus, N. Tazi, S. Pyysalo, T. Wolf, and C. Raffel. Scaling data-constrained language models. In *the coming Thirty-sixth Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=j5BuTrEj35
- T. Le Scao and A. Rush. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021. URL https://aclanthology.org/2021.naacl-main.208

Chapter 2

A primer on large language models

2.1 Some terminology

First, a bit of terminology is in order. For now, we have mostly been concerned with language models: as we briefly mentioned in the introduction, a language model is simply a probability distribution, a likelihood function, over the space of all possible text sequences. Language models come in several varieties depending on a number of features:

Model. Language models are trained estimators, who reproduce the statistical regularities of a training corpus. For this, several types of *models* have been used. The very first experiments, conducted by Shannon [Shannon, 1948], compared character n-gram models and human estimation. N-gram models stayed one of the main engines of speech decoding, the standard application of language modeling [Bahl et al., 1983]. One important distinction was between models that explicitly modeled syntax [Moore et al., 1996] to more purely statistical models [Brown et al., 1992] that operated solely on word counts and eventually prevailed. At the time, language models were still useful mostly as a component of speech transcription or machine translation, where they informed the probability of the output sequence. Neural methods eventually took over, with multi-layer perceptrons (MLP) [Bengio et al., 2000], then recurrent neural networks (RNNs) [Mikolov et al., 2010] first attaining state-of-the-art results. RNNs were then improved with the attention mechanism [Bahdanau et al., 2016], until an architecture based solely on attention, *transformers* [Vaswani et al., 2017], took over. Virtually all modern language models, and all those presented in this thesis, are transformers.

Objective. Several language modeling *objectives* coexist. To make estimation tractable, likelihood is always decomposed word-by-word as the likelihood of each element given a certain *context*.

- *Causal* language models, also called *autoregressive*, estimate the likelihood of each word in a sentence one by one in order. The likelihood of the total sequence then has a simple form as the product of the likelihoods of individual words given only previous context. This is particularly suited to text *generation*: one can *decode* from a model by estimating the most likely next word, or group of words, step by step until the sequence ends.

- *Masked* language models start from a full sequence from which a word, or group of words, has been removed. The objective is to estimate the likelihood of the possible words that could fill the void. Because they allow for information to be shared in both directions (later words can influence the choice of earlier ones), they are not practical for generation but are typically used for *representation*: embedding a text sequence into vector space.

- Although those two are the most useful, many others have been tried, often to join the strengths of both. The most important are *sequence-to-sequence* objectives, where text is explicitly separated into an input, which is read, and an output that is usually a response to it rather than a continuation. In contrast, for pure autoregressive models, output continuously becomes input, and the two are not distinguished. This also allows the input to be read in both directions if desired, and then for the output to be generated autoregressively.

Size and compute We have mentioned large language models (LLMs) a few times already: models that exceed a certain number of parameters. However, this limit number is a moving target that increases with time: calling a model large mostly means "hard to reproduce for most institutions". As the amount and availability of computing power inexorably increases with time, models fall out of this window. At the time of writing, most practitioners estimate that large language models start between one billion and 10 billion parameters [Neubig, 2023]. A last term we would like to introduce is *compute*[1], the abstract measure of an amount of computation, often instantiated as a simple count of the floating-point operations that went into training a model. As LLMs scale with compute and not strictly with parameter size, this focus on size can be misleading - a 10 billion parameter model trained for 10^{23} operations will outdo a 100 billion parameter model trained on 10^{22}. We will often refer to the *compute budget*, that is, the amount of compute earmarked for training, within the constraints of which practitioners work to train the best final model.

2.2 How did we get there?

2.2.1 Pretrained models

From their origin as a part of speech recognizer and machine translation systems, language models have historically been components for other tasks. In some sense, this is because the core task of estimating probability distributions over a sequence of text, even if it is the most fundamental statistical inference over language, is not that interesting by itself. Even though it is the core part of text generation, text generation by itself would only be a reflection of the contents of the training data; generation is almost always *conditioned*. This can be understood in the probabilistic sense, that is, the output sequence has to be the most likely sequence from the modeled distribution conditioned on some input. In this sense, pretrained models, that is statistical models trained as pure language models and then used for some other purpose, have always been around.

They were at the time, however, confined to generative NLP tasks such as summarization or the aforementioned speech transcription and machine translation. The use of pretrained models for every NLP task, including non-generative tasks such as classification or retrieval, starts with [Mikolov et al., 2013], an unsupervised algorithm to learn context-independent vector features for each word in a corpus. Word2vec was the first pretraining paper in the sense that the training task, a form

[1]Used here as a noun, in a return to the medieval English use, rather than a verb as usually.

of masked language modeling, was just a means to an end; instead, the goal was to learn vector representations for text in the process of optimizing that objective. Those representations, called word vectors, were then usable by other practitioners as input for their tasks of interest, thus decoupling learning representations for language — for which immense corpora of unstructured text are available — from learning a task — for which structured data, rarer, is needed.

At the same time, independently, pretraining was causing a paradigm shift in computer vision, with convolutional neural networks trained on ImageNet, a classification benchmark [Russakovsky et al., 2014], then adapted to other uses [Girshick et al., 2013, Donahue et al., 2013], yielding large advances. Explicitly inspired by the Imagenet moment in computer vision, ULMFiT [Howard and Ruder, 2018], a pretrained causal RNN, then GPT and BERT, causal and masked transformers respectively [Radford, 2018, Devlin et al., 2018], were the first models to bring modern pretraining to NLP. In this method, as opposed to word2vec and its context-dependent successors [Dai and Le, 2015], the same model is used at all stages of the adaptation, rather than just as a feature extraction layer that fed into a task-specific architecture. After pretraining, the model can be adapted to an arbitrary downstream task by replacing the final word prediction layer with a linear classifier, and then finetuning on task-specific data.

If word2vec was akin to learning the meaning of every word before learning to solve language tasks, those models first learned to speak English, then could be adapted to any downstream task - much easier than learning to summarize and learning to speak English at the same time. This technique became quickly available in other languages, first with a wave of BERT-for-X models [Le et al., 2020, Chan et al., 2020, Cañete et al., 2020], then with the apparition of the first massively multilingual models, trained on large amounts of data from a wide variety of different languages [Lample and Conneau, 2019, Conneau et al., 2020], which will be one of the main subjects of this thesis. The objective of multilingual language models was two-fold: yielding cross-lingual representations, which could carry over from language to language, and improving performance on the lowest-resource languages, with the quality of the high-resource representations carrying over thanks to transfer between languages.

Powered also by infrastructure that allowed practitioners to share and re-use base models and their finetuned versions, most notably the `transformers` library from Hugging Face, the "Github of AI", pretraining-then-finetuning quickly became the default solution for most of NLP. In 2021, 46.7% of the ACL papers cited BERT, and 20% cited Hugging Face. [Gururaja et al., 2023]

2.2.2 Zero-shot

A general principle of AI research, customarily called the Bitter Lesson[2] [Sutton, 2019] is that as compute becomes exponentially more available with time, general-purpose methods that leverage computation efficiently are eventually favored over methods that take in human input and priors, in the same way that purely statistical n-gram language models eventually prevailed against syntactically-inspired competitors at the very start of this chapter.

Thanks to recent innovations such as subword tokens [Schuster and Nakajima, 2012] — where the units of text passed to the model are drawn from a much smaller vocabulary of subwords, rather than the entire vocabulary of words — and the computing efficiency of transformers, those models were even smaller than word vector dictionaries (1.2GB for BERT-large, 2GB for word2vec) and not much

[2]Also sometimes called Jelinek's principle, after Frederick Jelinek's famous quip: "Any time a linguist leaves the group the recognition rate goes up."

harder to train. Transformer language models were uniquely suited to large-scale parallel computation over clusters of specific hardware, with their large matrix multiplications without recurrence and their simple, fundamental learning objective. Pretrained models yielded predictable increases in performance with additional compute, with BERT-large outdoing BERT-base, ROBERTA, trained for longer, outdoing BERT [Liu et al., 2019]...

In parallel, a nascent field of research was concerned with getting multi-task capabilities out of a single model, in the hope that more general single models would be able to solve a wide gamut of NLP tasks. In particular, a technique used to adapt models to a diverse set of tasks was giving them instructions in natural language to explain each of them [Trinh and Le, 2018, McCann et al., 2018], in the hope that they would pick up on the similarity with similar examples in the training data. McCann et al. [2018] finetuned on a few prompted examples to warm-start the process. Trinh and Le [2018] limited itself to a single task, the Winograd schema challenge, a hard commonsense reasoning problem [Levesque et al., 2012]. However, by training for the first time on a very large, diverse corpus with content pulled from the wider web, they ensured that the necessary information would already be in the model without finetuning. Making use of unprecedented amounts of compute and diverse web data, GPT-2 [Radford et al., 2019] was the first language model to be called large, and the first to demonstrate adaptation to multiple new tasks in this manner without any finetuning.

This absence of need for task-specific data, dubbed *zero-shot* classification after a similar concept in computer vision, was a strong demonstration of the power of generalization of language models. After all, the ability of humans to perform a novel task only from instructions, without examples, still set them apart from automated systems. The field of scaling laws emerged to predict final performance as a function of high-level inputs such as total number of training operations, size of the model, and amount of training data [Kaplan et al., 2020, Henighan et al., 2020, Aghajanyan et al., 2023]. Its results, predicting that performance would keep growing smoothly as a power law of the invested compute, set the stage for a large bet on this potential. The result was GPT-3 [Brown et al., 2020], the first language model of industrial scale and industrial interest. For the first time, on many tasks, zero-shot performance was within reach of the finetuned state-of-the-art. GPT-3 also showed completely novel generalization abilities, notably *in-context learning*: it could use labeled demonstrations to help it perform a task simply by feeding them in the context before the input, without the need to modify weights through finetuning. After this point, zero-shot, and in-context-learning-powered few-shot, as opposed to finetuning, became the main metric to evaluate state-of-the-art language models.

2.2.3 Instruction following

The zero-shot prompts the community had been using at the time had been tailored for the causal language modeling objective. The classification power of prompts was understood to be a natural extension of the language modeling objective, and their goal was thus to nudge the model toward the state, the mode in the distribution of training data, that had seen this before and would give the correct answer with the most probability at the end of the sentence. This had the effect of separating prompting from its earlier acception as a human-comprehensible technique. This distinction can be illustrated by a prompting spectrum of datasets and techniques:

- At one extreme, P3 [Sanh et al., 2022] (extended in xP3 [Muennighoff et al., 2022b]) was a set of prompts for a large variety of tasks crowdsourced from participants in the BigScience workshop, domain experts in their respective tasks and languages, but not prompting experts. The result was a set of instructions close to those you would use to explain the task to a human.

- The GPT-3 paper [Brown et al., 2020] published the list of prompts that had been used to showcase the zero-shot capabilities of the model. Although still human-readable, of course, those had been chosen to optimize zero-shot performance in an autoregressive language model. Enshrined in various LLM evaluation frameworks, most notably the Eleuther AI LLM Harness [Gao et al., 2021], they were the first benchmark to evaluate zero-shot performance.

- In order to optimize further for performance, the field of prompt tuning developed beyond the need for human readability: systems such as Autoprompt [Shin et al., 2020] used gradient descent to find prompts that would optimize performance on certain tasks and benchmarks, and found that nonsensical instructions[3] usually lay at the bottom of the search landscape.

- Finally, at the other extreme, prefix tuning [Li and Liang, 2021, Lester et al., 2021] was a technique to find "soft prompts", task-specific vectors in embedding space. The model would then read the sequence prepended with the soft prompt and output the answer to the task.

However, widespread adoption of language models could not come without the human-comprehensible aspect of instructions, and adaptability to novel, fuzzily-defined tasks. If the goal is to have a layman use a model to accomplish a new task, it is conceivable to have them spend a bit of time to find the right prompt, but not to apply gradient descent. To solve this issue, a first solution was *multi-task finetuning*. Introduced in T0 [Sanh et al., 2022] and FLAN [Wei et al., 2021a], this technique involved an extra round of finetuning on a set of human-written prompted tasks such as P3: if causal language models imitate their training data, we can make a model more functional by shifting its training distribution closer to its final use. This allowed T0 models to top the existing benchmarks, and proved that language models could generalize, as multi-task finetuned models yielded better performance on tasks not seen at finetuning, and even languages seen at pretraining and not at finetuning [Muennighoff et al., 2022b], a big draw of multilingual LMs.

Those prompting choices were all driven by the choice of the causal language modeling objective. For example, Wang et al. [2022a] found that although starting from an encoder-decoder model trained with a sequence-to-sequence prevailed over decoder-only causal language models, the difference was much larger on P3 than on the GPT-3 prompts. In order to allow for complete human-controllability of language models, then, a solution was to change the training objective. InstructGPT [Ouyang et al., 2022] used PPO [Schulman et al., 2017], a reinforcement learning objective, to get models to follow instructions, the success of which would eventually to the release of ChatGPT and the development of a broader field of reinforcement learning from human feedback. Since then, a whole industry dedicated to creating and curating human feedback and annotations has appeared, and competing objectives, such as DPO [Rafailov et al., 2023] are an active field of research, with variants of language modeling still going strong thanks to their simplicity.

[3]"<input> atmosphere alot dialogue Clone totally <output>" was a top performer for sentiment analysis.

2.3 What do we expect from them?

2.3.1 Evaluation benchmarks

The most direct way to track the expectations of the NLP and broader machine learning community for language models is to look at the evolution of the different benchmarks that have been used to measure progress. Benchmark creation is an underrated and resource-consuming endeavor, and NLP benchmarks tend to be replaced slowly and to integrate mostly existing tasks or problems, often repurposed for new testing frameworks.

During the pretrained era, the dominance of BERT, then the race between BERT and its later siblings, was measured on pre-existing small multi-task benchmarks, usually of a small dozen or ten academic NLP tasks with their assorted data, which was all the training data available for earlier methods and served as finetuning data now. ULMFiT, BERT, and GPT had all only been evaluated on a handful of tasks. When GLUE [Wang et al., 2018] came out, with its 11 well-known tasks, it quickly became the benchmark of reference. After a year and a half, progress in the pretraining-then-finetuning paradigm had made progress difficult to evaluate on GLUE, with leader Yang et al. [2019] nearly on par with human performance, and surpassing it even on one of the tasks. SuperGLUE [Wang et al., 2019b] was released afterward, with novel tasks crowdsourced by the community in a contest, but lasted only for a year, with DeBERTa [He et al., 2020] reaching human performance.

This meant that the rise of zero-shot prompting as a possible evaluation method for language models was embraced by a large part of the community as a way to keep moving forward. Even after GPT-3 was released, state-of-the-art models seemed to need lots of additional work to reach superhuman performance in zero-shot. This style of evaluation, however, came with lots of noise: models tended to get all the points or none per task, making for high variance, signal was low as performance was still struggling, and evaluation was highly sensitive to the choice of prompt. [Jiang et al., 2020]. This led to wide collaborative efforts to pool resources together and create benchmarks of hundreds of tasks, with well-identified prompts [Srivastava et al., 2022, Liang et al., 2022a]. Since then, language models have entered commercial, real-world use, where tasks are novel most of the time, and fuzzily defined. The rise of multi-task finetuning has raised concerns that instruct-tuned models were optimizing for a direction, mostly made of academic tasks, that was orthogonal to real-world use. Indeed, with help from the latest multi-task datasets [Mukherjee et al., 2023], it is possible to create models with arbitrarily high scores on the benchmarks, but that does not seem to translate into increased real-world use. Zero-shot benchmarks are still the measure of reference for pure pretrained models, especially hard, well-known tasks such as MMLU or ARC [Hendrycks et al., 2021, Clark et al., 2018]. For finetuned instruct models, the goal is now controlled open-ended generation in answer to fuzzy user queries. Although existing datasets are still rather small, tasks such as MT-Bench [Zheng et al., 2023] and AlpacaEval [Li et al., 2023b] have allowed the community to test their models in this setting, bypassing the difficulty of getting human ratings for open-ended answers by querying state-of-the-art language models, which correlates well with human judgments.

2.3.2 A platform for all modalities

For now, we have only mentioned models trained on natural language. Although phenomena in many other modalities can be described in text, LMs are not by themselves the right models to treat other types of data. For a variety of reasons, however - the high availability of language pretraining data; simplicity of the autoregressive objective; established models, understanding, and infrastructure; evidence that natural language pretraining transfers particularly well to other types of data, making it a privileged modality of sorts for training transformers - work in several other modalities starts from existing language models, or from the language modeling formula, to adapt LMs directly to a new modality. Most notable are the finetuning of LLMs for code generation [Chen et al., 2021, Rozière et al., 2023] or mathematics [Lewkowycz et al., 2022] and the emergence of massively multimodal autoregressive models [Aghajanyan et al., 2022, Reed et al., 2022].

Expanding beyond the scope of pure autoregressive language models, however, a rich body of work endows LLMs with capabilities in other modalities by joining them with distinct sensors or actuators in new domains, such as vision input [Jia et al., 2021, Alayrac et al., 2022, Laurençon et al., 2023] or output [Ramesh et al., 2021a, Yu et al., 2022], speech input [Gong et al., 2023, Fathullah et al., 2023], game action inputs and outputs [MFAIRD, 2022], or even robot action outputs [Driess et al., 2023]. In a sense, this goes back to earlier work such as image captioning [Young et al., 2014] or the aforementioned use of language models for the output of speech recognizers. In this new framework, however, LMs are not just output layers: they are the central system where most of the computation happens, interpreting e.g. commands and vision input into robot policies in the case of Driess et al. [2023].

2.3.3 So what do we want language models for?

The goal of any machine learning algorithm, regardless of the model getting optimized, is generalization. Once a model is trained to perform a task on a dataset, can it perform in other settings? In classical machine learning, this was often expressed in terms of train and test sets and distribution shifts: does performance decrease on hidden data points drawn from the same distribution? The zero-shot capabilities of LLMs seem astonishing in this paradigm, most notably dealing with new words or concepts without retraining if the context contains an explainer[4], generalizing from the language modeling objective to entirely new tasks. Of course, there is nothing magical about those results: as was already the case in the first zero-shot papers [Trinh and Le, 2018, Radford et al., 2019], one of the main ingredients of LLM training is compiling a dataset so large that anything that is asked from the model is an interpolation of elements it has already seen.

This ability to deal with a whole range of first-time, fuzzily defined tasks, with a natural language interface that does not require special technical training, is currently the main draw of LLMs, and the source of the excitement around them in the world at large beyond the NLP community. We then need abundant, diverse, high-quality data, and models powerful enough to be able to generalize from it. Those needs, for now, are mostly met in English. The spirit of this dissertation will be to try to make some progress in languages with less abundant resources.

[4]The success of the "Farduddle test" from GPT-3, where the model has to complete the sequence "To do a ’farduddle’ means to jump up and down really fast. An example of a sentence that uses the word farduddle is:" was an eye-opener for the author.

Chapter 3

Large-scale multilingual data collection

3.1 Introduction

As language models grow ever larger, so does the need for large-scale high-quality text datasets, especially in multilingual settings. In order to train language models, however, one must first have the language data to feed them. This chapter documents the data creation and curation efforts undertaken within Bigscience, an internal research collaboration, to assemble the Responsible Open-science Open-collaboration Text Sources (ROOTS) corpus, a 1.6TB multilingual dataset that was used to train the family of BLOOM [BigScience Workshop, 2022] models, which will be presented next chapter. The corpus, and the tools used to create it, were all made publicly available.

We describe our approach to curating a web-scale dataset covering 59 languages, 46 natural languages and 13 programming languages — given the importance we placed on language expertise, the language choice was chiefly driven by the communities who participated in the effort. Our final corpus is made up of two main components: 62% of the text comes from a community-selected and documented list of language data sources, and 38% consists of text extracted from a pre-processed web crawl, OSCAR [Ortiz Suárez et al., 2020], filtered with the help of native speakers.

Our goal when building ROOTS was to empower large-scale monolingual and multilingual modeling projects. This has two aspects here: describing the creation process and final composition of ROOTS, and showcasing the numerous data tools[1] that were developed along the way and enabled us to curate, source, clean and inspect all 498 constituent datasets that come together to constitute ROOTS. Our approach is meant to encourage practitioners to re-use our data corpus and provide them with all of the necessary knowledge about its components, but also to make it easier for them to create their own corpora, especially in under-served languages.

3.2 Related Work

Large Language Models and Large Text Corpora The current dominant paradigm in natural language processing relies heavily on pre-trained models: large language models that can then be fine-tuned on a downstream task [Howard and Ruder, 2018, Devlin et al., 2018] or even used as-is without additional data [Radford et al., 2019, Brown et al., 2020]. In this paradigm, performance

[1] https://github.com/bigscience-workshop/data-preparation

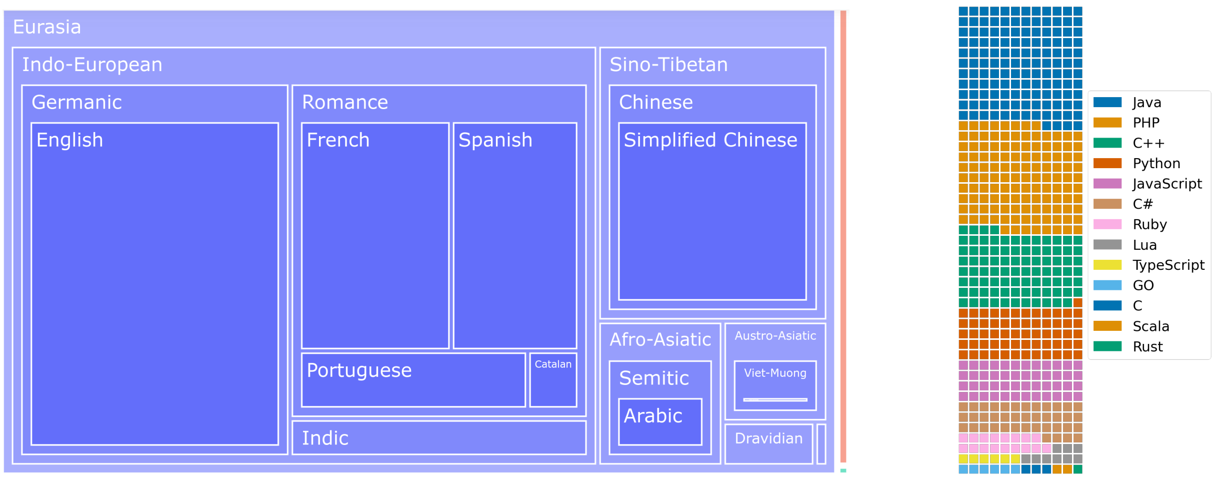


FIGURE 3.1: Overview of ROOTS. Left: A treemap of natural language representation in number of bytes by language family. The bulk of the graph is overwhelmed by the 1321.89 GB allotted to Eurasia. The orange rectangle corresponds to the 18GB of Indonesian, the sole representative of the Papunesia macroarea, and the green rectangle to the 0.4GB of the Africa linguistic macroarea. Right: A waffle plot of the distribution of programming languages by number of files. One square corresponds approximately to 30,000 files.

is directly correlated on both the model size and the dataset size and quality [Kaplan et al., 2020], with recent models trained on up to 1.4 trillion tokens [Hoffmann et al., 2022a] and dataset creation pipelines representing a significant part of large language model projects. Most such datasets, however, are not released, hindering further research. Exceptions include the Pile [Gao et al., 2020a], a curated corpus of datasets for language modeling that has become widely used for training state-of-the-art English-language models [Lieber et al., 2021, Smith et al., 2022, Black et al., 2022, Zhang et al., 2022], and C4 and mC4 [Raffel et al., 2020, Xue et al., 2020a], which have powered the T5 family of models; CC100 [Conneau et al., 2020] which has seen heavy use for multilingual modeling; and OSCAR [Ortiz Suárez et al., 2019], which has enabled monolingual non-English models.

Tooling, Visualization, and Replication Upstream from the finalized training datasets is the issue of processing methods and pipelines: both the operations that the datasets go through and the engineering effort required to apply them at terabyte scales. Existing work tends to fall on a spectrum from no details at all [Brown et al., 2020] to detailed filtering instructions, with [Raffel et al., 2020] or without the dataset release [Rae et al., 2021b] to detailed filtering instructions with the accompanying code [Gao et al., 2020a, Conneau et al., 2020, Ortiz Suárez et al., 2019]. Even when the code is released, it tends to be built and tailored for the project's purpose. Consequently, large projects that do not re-use an existing dataset outright usually build their own pipeline rather than re-use an existing one on new data. However, data tools that were built and packaged in order to be used for other projects exist, such as OSCAR's Ungoliant and Goclassy [Abadji et al., 2021, Ortiz Suárez et al., 2019], which provides a distributed Common Crawl processing pipeline; CCNet [Wenzek et al., 2020], built for quality filtering of multilingual Common Crawl dumps; and OpenWebText [Gokaslan and Cohen, 2019], enabling Reddit dump processing.

Documenting Textual Corpora in NLP An inspiration for our work is a recent emphasis on a more in-depth documentation of what is included and what is not in the corpora used for training NLP models. The most notable example of this is the Pile, for which the authors themselves analyze and document a variety of syntactic and semantic properties of the dataset including structural statistics (n-gram counts, language, document sizes), topical distributions across its components, social bias and sentiment co-occurrence, pejorative content, and information about licensing and authorial consent, in addition to releasing a datasheet [Biderman et al., 2022]. Other LM pre-training datasets that have been documented and analyzed include C4 [Dodge et al., 2021, Luccioni and Viviano, 2021, Kreutzer et al., 2022], OSCAR [Kreutzer et al., 2022] and BookCorpus [Bandy and Vincent, 2021]. While this kind of documentation is far from standard practice, it is becoming increasingly common given recent calls for better documentation [Rogers, 2021, Bender et al., 2021] as well as empirical studies on data memorization in language models [Carlini et al., 2019, 2022].

3.3 (Crowd) Sourcing a Language Resource Catalogue

The first part of our corpus, accounting for 62% of the final dataset size (in bytes), was made up of a collection of monolingual and multilingual language resources that were selected and documented collaboratively through various efforts of the BigScience Data Sourcing working group. The first such effort consisted in creating a tool to support metadata collection through open submissions, called the BigScience Catalogue and running a series of hackathons in collaboration with locally-focused ML and NLP communities such as Masakhane, Machine Learning Tokyo and LatinX in AI where participants could add and document entries for their languages to the catalogue [McMillan-Major et al., 2022]. This yielded a set of 252 sources, including at least 21 per considered language category. We focused on metadata collection as a way to support selection of the sources for the final dataset and documentation of the final dataset. In parallel, working group participants gathered additional Arabic language resources in the Masader repository [Alyafeai et al., 2021], and proposed a list of websites of interest to increase the geographical diversity of our English, Spanish, and Chinese language data. Finally, in order to explicitly test large language models' ability to handle computer code along with natural language, we selected code data available on GitHub and StackExchange.

3.3.1 Obtaining Data from the Identified Resources

Gathering Identified Datasets and Collections. First, we leveraged the BigScience Catalogue

and the Masader repository to start obtaining text from identified sources, which included both existing NLP datasets and collections of documents of various compositions. Given the diversity of sources, hosting methods, data custodians, and formats, collecting this text required a collaborative effort. To that end, we established a 2-phase approach: first, collect as many data sources as possible in an easily accessible location; second, map all of them to a common format to ease further processing.

In the first phase, we organized an open hackathon to start gathering identified sources on the Hugging Face Datasets hub [Lhoest et al., 2021], in a dedicated organization[2] (in order to manage access controls). In the second phase, the collected datasets were furthered processed via (1) *Language segmentation*, whereby data sources were split using metadata for each covered language in order to

[2]https://hf.co/bigscience-catalogue-data

obtain monolingual datasets, and the use of (2) *Uniform interface* whereby a document consisted of two fields: "text" for the actual text content, and "meta" with a JSON representation of metadata for a given document, containing sufficient information to trace documents back to their original sources.

Table 3.3.1 presents all of the resulting data sources.

Dataset	Language	Source
AraBench	ar	Sajjad et al. [2020]
1.5 billion words Arabic Corpus	ar	El-Khair [2016]
BanglaLM	bn	Kowsher et al. [2021]
bangla sentiment classification datasets	bn	Rahman et al. [2018]
Question answering in Bengali	bn	Mayeesha et al. [2020]
Binhvq News Corpus	vi	Bình [2021]
Books by Book Dash	en, fr, xh, zu	https://bookdash.org/books/
Bloom Library	ak, bm, fon, ki, lg, ln, nso, rw, st, sw, tn, ts, xh, zu	bloomlibrary.org
BRAD 2.0	ar	Elnagar et al. [2018]
brWaC Corpus	pt	
EusCrawl	eu	Artetxe et al. [2022]
Catalan General Crawling	ca	Armengol-Estapé et al. [2021]
Catalan Government Crawling	ca	Armengol-Estapé et al. [2021]
Catalan Textual Corpus	ca	Armengol-Estapé et al. [2021]
Data on COVID-19 News Coverage in Vietnam	vi	Vuong et al. [2021]
DuReader	zhs	He et al. [2018]
Enriched CONLLU Ancora for ML training	ca	Rodriguez-Penagos and Armentano-Oller [2021a]
ESTER	fr	Galliano et al. [2006]
Github	code	Github on BigQuery
Habibi	ar	El-Haj [2020]
HAL	fr	hal.archives-ouvertes.fr/
IIT Bombay English-Hindi Parallel Corpus	hi	Kunchukuttan et al. [2018]
IndicNLP Corpus	bn, gu, hi, kn, ml, mr, or, pa, ta, te	Kunchukuttan et al. [2020]
Indo4B BPPT	id	Budiono et al. [2009]
Indo4B OPUS JW300	id	Agić and Vulić [2019]
Indo4B Kompas	id	Sakti et al. [2008]
Indo4B Parallel Corpus	id	Pisceldo et al. [2009]
Indo4B TALPCo	id	Nomoto et al. [2018]
Indo4B Tempo	id	Sakti et al. [2008]
Indonesian Frog Storytelling corpus	id	Moeljadi [2012]
Indonesian News Articles Published at 2017	id	Ashari [2018]
Indonesian News Corpus	id	Rahutomo and Miqdad Muadz Muzad [2018]
IndoNLI	id	Mahendra et al. [2021]
Indosum	id	Kurniawan and Louvan [2018]
KALIMAT	ar	El-Haj and Koulali [2013]
KSUCCA	ar	Alrabiah et al. [2013]
LABR	ar	Aly and Atiya [2013]
Language modeling data for Swahili	sw	Shikali and Refuoe [2019]
Leipzig Corpora Collection	ur	Goldhahn et al. [2012]
Mann Ki Baat	bn, gu, hi, ml, mr, or, ta, te, ur	Siripragada et al. [2020]
Masakhaner	ig, lg, rw, sw, wo, yo	Adelani et al. [2021]
MultiUN v2	en, ar, es, fr, zhs	Chen and Eisele [2012]
Opensubtitles2016	ca, en, ar, es, eu, fr, id, bn, hi, ml, ta, te, ur, pt, vi, zhs	Lison and Tiedemann [2016]
OpenITI proc	ar	Belinkov et al. [2019]
OPUS-100	ca, ar, eu, id, as, bn, gu, hi, ig, kn, ml, mr, or, pa, rw, ta, te, ur, pt, vi, xh, yo, zu	Zhang et al. [2020a]
ParlamentParla	ca	Külebi [2021]
PIB	bn, gu, hi, ml, mr, or, pa, ta, te, ur	Siripragada et al. [2020]
Project Gutenberg	en, es, fr, pt, zhs	gutenberg.org
QED (formely AMARA Corpus)	ar, en, es, fr, hi, pt, zhs, zht	Abdelali et al. [2014]
Recibrew	id	Wibowo [2020]
The Royal Society Corpus	en	Kermes et al. [2016]
S2ORC	en	Lo et al. [2020]
Samanantar	as, bn, gu, hi, kn, ml, mr, or, pa, ta, te	Ramesh et al. [2021b]
SANAD	ar	Einea et al. [2019]

Dataset	**Language**	**Source**
SciELO	en, es, pt	scielo.org
Stack Exchange	code	Gao et al. [2020a]
Swahili News Classification Dataset	sw	David [2020]
Tashkeela	ar	Zerrouki and Balla [2017]
TeCLa	ca	Carrino et al. [2021]
WIT^3	ca, ar, en, es, eu, fr, id, as, bn, gu, hi, kn, ml, mr, pa, sw, ta, te, ur, pt, vi, zhs	Cettolo et al. [2012]
The Pile: EuroParl	en, es, fr, pt	Gao et al. [2020a]
The Pile: USPTO	en	Gao et al. [2020a]
UIT-VSMEC	vi	Ho et al. [2020]
United Nations Parallel Corpus	ar, en, es, fr, zhs	Ziemski et al. [2016]
Unsupervised Cross-lingual Representation Learning at Scale Common Crawl Corpus	ne	Conneau et al. [2020]
Urdu Monolingual Corpus	ur	Jawaid et al. [2014]
VietAI SAT	vi	Ngo and Trinh [2021]
Vietnamese Poetry Corpus	vi	Nguyen et al. [2021]
UIT-VSFC	vi	Nguyen et al. [2018]
VilaQuAD	ca	Rodriguez-Penagos and Armentano-Oller [2021b]
VinBigdata-VSLP ASR Challenge 2020	vi	institute.vinbigdata.org/events/vinbigdata-chia-se-100-gio-du-lieu-tieng-noi-cho-cong-dong/
VinBigdata-VSLP Monolingual Corpus 2020	vi	institute.vinbigdata.org/events/vinbigdata-chia-se-100-gio-du-lieu-tieng-noi-cho-cong-dong/
VinBigdata-VSLP Bilingual Corpus 2020	vi	institute.vinbigdata.org/events/vinbigdata-chia-se-100-gio-du-lieu-tieng-noi-cho-cong-dong/
ViquiQuAD	ca	Rodriguez-Penagos and Armentano-Oller [2021c]
VNTQcorpus(big)	vi	http://viet.jnlp.org/download-du-lieu-tu-vung-corpus
Wikibooks	ca, ar, en, es, eu, fr, id, bn, hi, ml, mr, pa, ta, te, ur, pt, vi, zhs	wikibooks.org
Wikimedia	ca, id, hi, pt	wikimedia.org
Wikinews	ar, ca, en, es, fr, ta, pt, zhs	wikinews.org
Wikipedia	ak, ar, as, bm, bn, ca, en, es, eu, fr, id, ig, gu, hi, ki, kn, lg, ln, ml, mr, nso, ny, or, pa, pt, rn, rw, sn, st, sw, ta, te, tn, ts, tum, tw, ur, vi, wo, yo, zhs, zht, zu	wikipedia.org
Wikiquote	ar, ca, en, es, eu, fr, id, gu, hi, kn, ml, mr, ta, te, ur, pt, vi, zhs	wikiquote.org
Wikisource	ar, ca, es, eu, fr, id, as, bn, gu, hi, kn, ml, mr, or, pa, ta, te, pt, vi	wikisource.org
Wikiversity	ar, en, es, fr, hi, pt, zhs	wikiversity.org
Wikivoyage	en, es, fr, bn, hi, pt, vi, zhs	wikivoyage.org
Wiktionary	ar, ca, en, es, eu, fr, id, as, bn, gu, hi, kn, ml, mr, or, pa, ta, te, ur, pt, vi	wiktionary.org
WuDaoCorpora	zhs	Yuan et al. [2021]
XQUAD-ca	ca	Armengol-Estapé et al. [2021]

TABLE 3.1: List of crowdsourced datasets.

Pseudo-Crawled Data. Of the various categories of language resources identified through the data sourcing effort, websites stood out as one that required a particular effort and dedicated pipeline. We decided to design such a pipeline based on "pseudo-crawling": that is, rather than crawling the websites ourselves, we retrieved pages corresponding to the target domain names from 18 snapshots archived by Common Crawl in 2020 and 2021 in Web ARChive (WARC) format [Mohr et al., 2008]. These domain names came from two main sources: the homepage field in the metadata of the 252 above-mentioned catalogue entries when available (192 in total), and the 456 websites proposed by participants asynchronously to improve the geographical diversity of our language sources; which yielded a total of 614 unique domain names after deduplication.

We collected URLs contained within those domains using the Common Crawl index. The index provides metadata for every document including the page URL, WARC filename and record offsets, fetch status, content MIME type, etc. We ran a query matching all documents that share the domain name with a seed using Amazon Athena on Common Crawl's columnar index[3]. 48 of the 614 initial seed domain names had no matches in the index and were therefore left out. Once we obtained the document metadata, we fetched the WARC records using HTTP range requests with the start and end byte offsets. Since HTML web pages constitute the largest portion of pages contained in the Common Crawl dumps, we decided to only extract text from HTML pages. Documents in other formats were filtered out, ie XML, PDF, etc. 27 domain names were additionally removed from the list at this stage as we had not retrieved any HTML pages for them.

To extract the text from the HTML pages, we first minified the HTML code. Minification is the removal of unnecessary characters from the source code of a website. Inspired by Aghajanyan et al. [2022], we removed from the DOM-HTML all the sub-trees contained in a *<script>*, *<style>*, *<header>*, *<iframe>*, *<footer>* and *<form>* tag as well as all the sub-trees associated with a *<body>*, *<div>*, *<p>*, *<section>*, *<table>*, *<ul>*, *<ol>* or *<dl>* tag whose textual content was less than 64 characters long. The text was then extracted from the nodes of this new DOM-HTML and all text nodes concatenated according to a set of rules inspired by what Common Crawl's processing. The overall procedure enabled us to obtain text datasets for 539 domain names.

GitHub Code. We collected a code dataset from BigQuery[4] using the same language selection as AlphaCode [Li et al., 2022]. The dataset was then deduplicated of exact matches and filtered for source files with between 100 and 200,000 characters, between 15-65% alphabetic characters, a max line length of 20-1000 characters, and a token length standard deviation of more than 3. Due to a bug in the pre-processing pipeline the dataset was also filtered for GPL licenses only.

Merging and Deduplicating Sources. After gathering and processing language data via the three pipelines outlined above, we took a final step to manually inspect, deduplicate, and make a further selection of the sources. First, we addressed dataset overlap we found by looking through our sources. For example: *OpenITI* was present in both its raw form as well as a processed version. Consensus was reached to choose the latter version. Non-trivial datasets overlap included *s2orc* [Lo et al., 2020], *Arxiv* [Clement et al., 2019] and the *PubMed Central* subset of the Pile [Gao et al., 2020a]. We also performed cross-pipeline dataset deduplication, removing the pseudo-crawled Wikipedia and GitHub

[3]https://commoncrawl.org/2018/03/index-to-warc-files-and-urls-in-columnar-format/
[4]"GitHub on BigQuery: Analyze all the open source code"

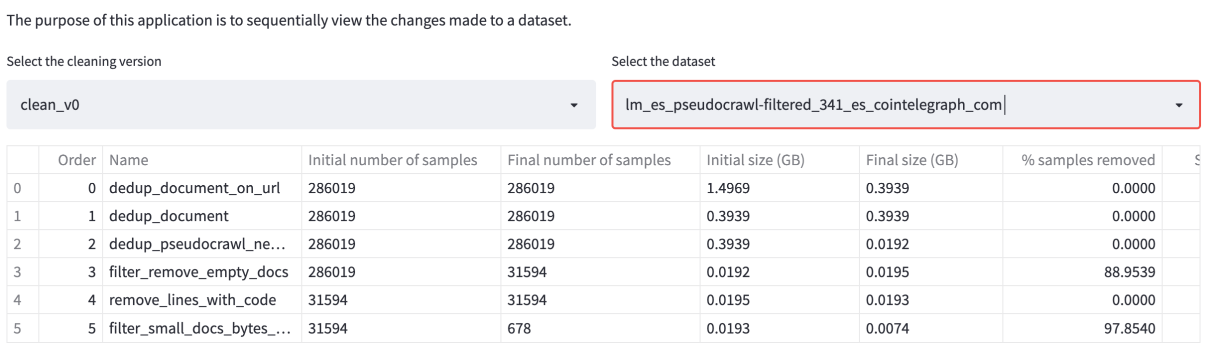
in favor of their other versions. We also removed datasets that we found had a high incidence of documents that were not fully in natural language (e.g. unexpected instances of SEO, HTML tags etc...), as well as very small datasets in the higher-resourced languages. Finally, pseudo-crawled sources were further processed to remove menus (with a heuristic consisting of removing lines that occurred in more than 1% of pages in a given domain) and pages that had a high incidence of character ngram repetition, low language identification confidence, or low proportion of closed class words (see Section 3.4). We then removed entire domains whose size was less than 2MB after this step, yielding 147 pseudo-crawl-based datasets, and a total of 517 datasets including all three pipelines.

3.3.2 Processing Pipeline for Quality Improvement on Crowdsourced Datasets

Once a text field was obtained, we attempted to improve the quality of that text. In the specific case of text extraction from HTML, we observe that not all text are relevant (menus, advertisements, repeated text on each page etc ...). In order to remove noisy data from our dataset, we applied a processing pipeline for each dataset consisting of a sequence of functions.

Functions were categorised as *document-scoped* or *dataset-scoped* functions. *Document-scoped* functions are operations that modify a document independently of other documents and *dataset-scoped* functions are operations that take into account the whole dataset. Orthogonal to this scope, functions were also separated into *cleaning* and *filtering* functions. *Cleaning functions* aim to remove text considered not part of the main document. Document-scoped cleaning functions can for example target leftover HTML tags. On the other end, dataset-scoped cleaning functions need the whole dataset to calculate a heuristic to determine how to modify each document. For instance, advertisements vary across datasets, making it harder to define a dataset-agnostic classifier for advertisement. Instead, we can index all the lines in a dataset and identify repeated lines on multiple pages as likely advertisements.*Filtering functions* aim at removing an entire document from the corpus. The reasons for choosing to remove a document completely are diverse: it may be because the document is considered to be of too poor quality, to be too complex to automatically fix or too similar to other examples already present in the corpus. The latter case, deduplication of a document, is dependent on whether an equivalent document already exists somewhere else in the dataset and is thus necessarily a dataset-scope function. The notion of equivalent documents has been explored by Lee et al. [2022]. In this case we provide deduplication via metadata (urls, normalised urls) and via text (exact string matching).

As datasets came from heterogeneous sources with different properties, each needs its own set of processing functions to correspond to our definition of natural language documents. In order to support participants in deciding what functions to apply to which, we built and released a *streamlit*-based visualization tool. Screenshots of the tool are presented in figure 3.2. This rapid feedback loop enabled us to update the pipeline consequently in an iterative process to finetune each processing pipelines across datasets and languages with the input of native speakers. This resulted in 485 non-empty datasets.



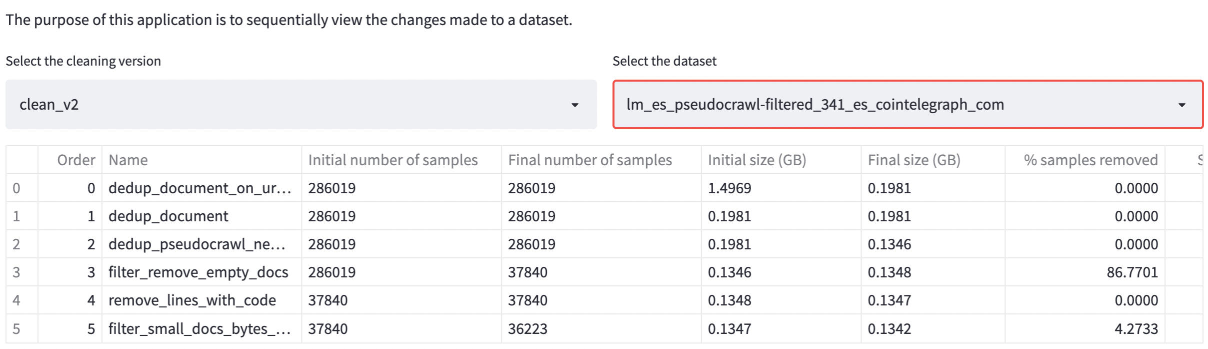
The purpose of this application is to sequentially view the changes made to a dataset.

Select the cleaning version: clean_v0

Select the dataset: lm_es_pseudocrawl-filtered_341_es_cointelegraph_com

	Order	Name	Initial number of samples	Final number of samples	Initial size (GB)	Final size (GB)	% samples removed
0	0	dedup_document_on_url	286019	286019	1.4969	0.3939	0.0000
1	1	dedup_document	286019	286019	0.3939	0.3939	0.0000
2	2	dedup_pseudocrawl_ne…	286019	286019	0.3939	0.0192	0.0000
3	3	filter_remove_empty_docs	286019	31594	0.0192	0.0195	88.9539
4	4	remove_lines_with_code	31594	31594	0.0195	0.0193	0.0000
5	5	filter_small_docs_bytes_…	31594	678	0.0193	0.0074	97.8540

(A) Pipeline v0



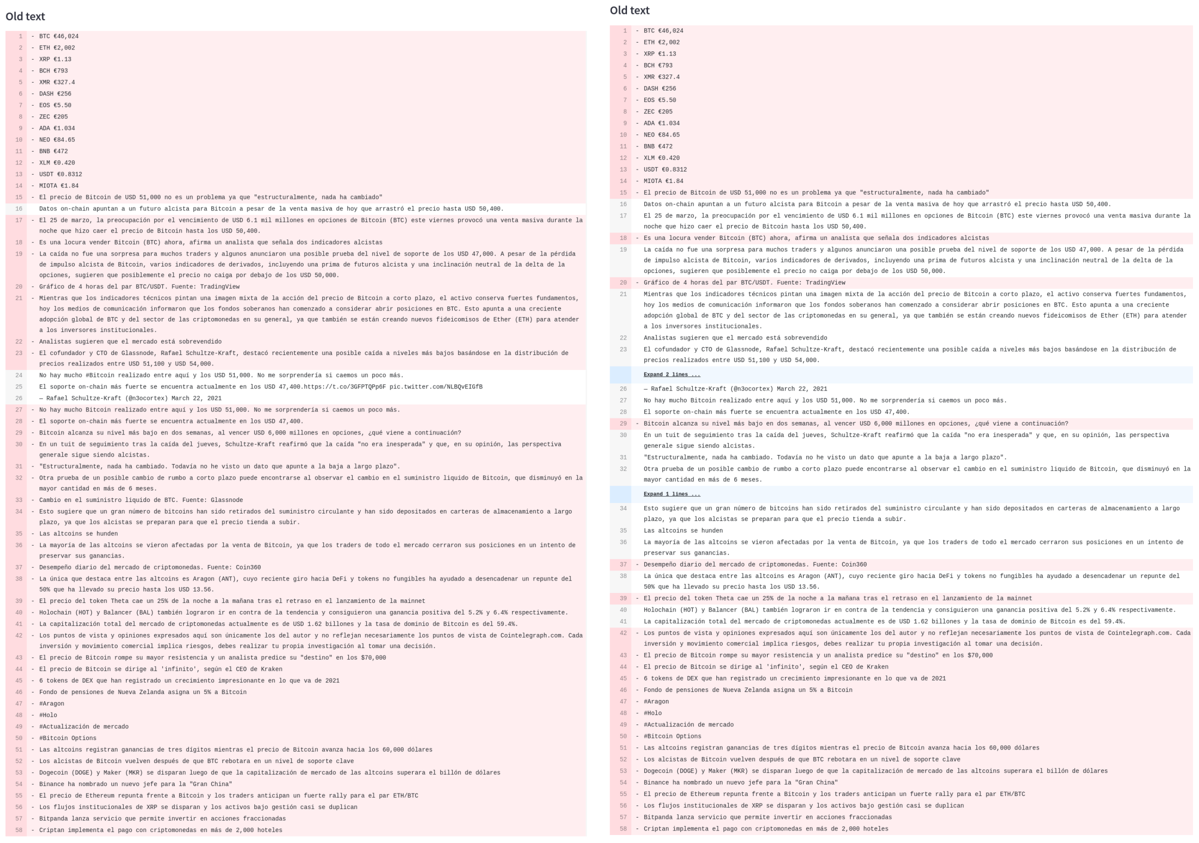
The purpose of this application is to sequentially view the changes made to a dataset.

Select the cleaning version: clean_v2

Select the dataset: lm_es_pseudocrawl-filtered_341_es_cointelegraph_com

	Order	Name	Initial number of samples	Final number of samples	Initial size (GB)	Final size (GB)	% samples removed
0	0	dedup_document_on_ur…	286019	286019	1.4969	0.1981	0.0000
1	1	dedup_document	286019	286019	0.1981	0.1981	0.0000
2	2	dedup_pseudocrawl_ne…	286019	286019	0.1981	0.1346	0.0000
3	3	filter_remove_empty_docs	286019	37840	0.1346	0.1348	86.7701
4	4	remove_lines_with_code	37840	37840	0.1348	0.1347	0.0000
5	5	filter_small_docs_bytes_…	37840	36223	0.1347	0.1342	4.2733

(B) Pipeline v2



Old text

```
- BTC €46,024
- ETH €2,002
- XRP €1.13
- BCH €793
- XMR €327.4
- DASH €256
- EOS €5.50
- ZEC €205
- ADA €1.034
- NEO €84.65
- BNB €472
- XLM €0.420
- USDT €0.8312
- MIOTA €1.84
- El precio de Bitcoin de USD 51,000 no es un problema ya que "estructuralmente, nada ha cambiado"
  Datos on-chain apuntan a un futuro alcista para Bitcoin a pesar de la venta masiva de hoy que arrastró el precio hasta USD 50,400.
- El 25 de marzo, la preocupación por el vencimiento de USD 6.1 mil millones en opciones de Bitcoin (BTC) este viernes provocó una venta masiva durante la
  noche que hizo caer el precio de Bitcoin hasta los USD 50,400.
- Es una locura vender Bitcoin (BTC) ahora, afirma un analista que señala dos indicadores alcistas
- La caída no fue una sorpresa para muchos traders y algunos anunciaron una posible prueba del nivel de soporte de los USD 47,000. A pesar de la pérdida
  de impulso alcista de Bitcoin, varios indicadores de derivados, incluyendo una prima de futuros alcista y una inclinación neutral de la delta de la
  opciones, sugieren que posiblemente el precio no caiga por debajo de los USD 50,000.
- Gráfico de 4 horas del par BTC/USDT. Fuente: TradingView
- Mientras que los indicadores técnicos pintan una imagen mixta de la acción del precio de Bitcoin a corto plazo, el activo conserva fuertes fundamentos,
  hoy los medios de comunicación informaron que los fondos soberanos han comenzado a considerar abrir posiciones en BTC. Esto apunta a una creciente
  adopción global de BTC y del sector de las criptomonedas en su general, ya que también se están creando nuevos fideicomisos de Ether (ETH) para atender
  a los inversores institucionales.
- Analistas sugieren que el mercado está sobrevendido
- El cofundador y CTO de Glassnode, Rafael Schultze-Kraft, destacó recientemente una posible caída a niveles más bajos basándose en la distribución de
  precios realizados entre USD 51,100 y USD 54,000.
  No hay mucho #Bitcoin realizado entre aquí y los USD 51,000. No me sorprendería si caemos un poco más.
  El soporte on-chain más fuerte se encuentra actualmente en los USD 47,400.https://t.co/3GFPTQPp6F pic.twitter.com/NLBQvEIGfB
  — Rafael Schultze-Kraft (@n3ocortex) March 22, 2021
- No hay mucho Bitcoin realizado entre aquí y los USD 51,000. No me sorprendería si caemos un poco más.
- El soporte on-chain más fuerte se encuentra actualmente en los USD 47,400.
- Bitcoin alcanza su nivel más bajo en dos semanas, al vencer USD 6,000 millones en opciones, ¿qué viene a continuación?
- En un tuit de seguimiento tras la caída del jueves, Schultze-Kraft reafirmó que la caída "no era inesperada" y que, en su opinión, las perspectiva
  generale sigue siendo alcistas.
- "Estructuralmente, nada ha cambiado. Todavía no he visto un dato que apunte a la baja a largo plazo".
- Otra prueba de un posible cambio de rumbo a corto plazo puede encontrarse al observar el cambio en el suministro liquido de Bitcoin, que disminuyó en la
  mayor cantidad en más de 6 meses.
- Cambio en el suministro liquido de BTC. Fuente: Glassnode
- Esto sugiere que un gran número de bitcoins han sido retirados del suministro circulante y han sido depositados en carteras de almacenamiento a largo
  plazo, ya que los alcistas se preparan para que el precio tienda a subir.
- Las altcoins se hunden
- La mayoría de las altcoins se vieron afectadas por la venta de Bitcoin, ya que los traders de todo el mercado cerraron sus posiciones en un intento de
  preservar sus ganancias.
- Desempeño diario del mercado de criptomonedas. Fuente: Coin360
- La única que destaca entre las altcoins es Aragon (ANT), cuyo reciente giro hacia DeFi y tokens no fungibles ha ayudado a desencadenar un repunte del
  50% que ha llevado su precio hasta los USD 13.56.
- El precio del token Theta cae un 25% de la noche a la mañana tras el retraso en el lanzamiento de la mainnet
- Holochain (HOT) y Balancer (BAL) también lograron ir en contra de la tendencia y consiguieron una ganancia positiva del 5.2% y 6.4% respectivamente.
- La capitalización total del mercado de criptomonedas actualmente es de USD 1.62 billones y la tasa de dominio de Bitcoin es del 59.4%.
- Los puntos de vista y opiniones expresados aquí son únicamente los del autor y no reflejan necesariamente los puntos de vista de Cointelegraph.com. Cada
  inversión y movimiento comercial implica riesgos, debes realizar tu propia investigación al tomar una decisión.
- El precio de Bitcoin rompe su mayor resistencia y un analista predice su "destino" en los $70,000
- El precio de Bitcoin se dirige al 'infinito', según el CEO de Kraken
- 6 tokens de DEX que han registrado un crecimiento impresionante en lo que va de 2021
- Fondo de pensiones de Nueva Zelanda asigna un 5% a Bitcoin
- #Aragon
- #Holo
- #Actualización de mercado
- #Bitcoin Options
- Las altcoins registran ganancias de tres dígitos mientras el precio de Bitcoin avanza hacia los 60,000 dólares
- Los alcistas de Bitcoin vuelven después de que BTC rebotara en un nivel de soporte clave
- Dogecoin (DOGE) y Maker (MKR) se disparan luego de que la capitalización de mercado de las altcoins superara el billón de dólares
- Binance ha nombrado un nuevo jefe para la "Gran China"
- El precio de Ethereum repunta frente a Bitcoin y los traders anticipan un fuerte rally para el par ETH/BTC
- Los flujos institucionales de XRP se disparan y los activos bajo gestión casi se duplican
- Bitpanda lanza servicio que permite invertir en acciones fraccionadas
- Criptan implementa el pago con criptomonedas en más de 2,000 hoteles
```

Old text

```
- BTC €46,024
- ETH €2,002
- XRP €1.13
- BCH €793
- XMR €327.4
- DASH €256
- EOS €5.50
- ZEC €205
- ADA €1.034
- NEO €84.65
- BNB €472
- XLM €0.420
- USDT €0.8312
- MIOTA €1.84
- El precio de Bitcoin de USD 51,000 no es un problema ya que "estructuralmente, nada ha cambiado"
  Datos on-chain apuntan a un futuro alcista para Bitcoin a pesar de la venta masiva de hoy que arrastró el precio hasta USD 50,400.
  El 25 de marzo, la preocupación por el vencimiento de USD 6.1 mil millones en opciones de Bitcoin (BTC) este viernes provocó una venta masiva durante la
  noche que hizo caer el precio de Bitcoin hasta los USD 50,400.
- Es una locura vender Bitcoin (BTC) ahora, afirma un analista que señala dos indicadores alcistas
  La caída no fue una sorpresa para muchos traders y algunos anunciaron una posible prueba del nivel de soporte de los USD 47,000. A pesar de la pérdida
  de impulso alcista de Bitcoin, varios indicadores de derivados, incluyendo una prima de futuros alcista y una inclinación neutral de la delta de la
  opciones, sugieren que posiblemente el precio no caiga por debajo de los USD 50,000.
- Gráfico de 4 horas del par BTC/USDT. Fuente: TradingView
  Mientras que los indicadores técnicos pintan una imagen mixta de la acción del precio de Bitcoin a corto plazo, el activo conserva fuertes fundamentos,
  hoy los medios de comunicación informaron que los fondos soberanos han comenzado a considerar abrir posiciones en BTC. Esto apunta a una creciente
  adopción global de BTC y del sector de las criptomonedas en su general, ya que también se están creando nuevos fideicomisos de Ether (ETH) para atender
  a los inversores institucionales.
  Analistas sugieren que el mercado está sobrevendido
  El cofundador y CTO de Glassnode, Rafael Schultze-Kraft, destacó recientemente una posible caída a niveles más bajos basándose en la distribución de
  precios realizados entre USD 51,100 y USD 54,000.
  Expand 2 lines ...
  — Rafael Schultze-Kraft (@n3ocortex) March 22, 2021
  No hay mucho Bitcoin realizado entre aquí y los USD 51,000. No me sorprendería si caemos un poco más.
  El soporte on-chain más fuerte se encuentra actualmente en los USD 47,400.
- Bitcoin alcanza su nivel más bajo en dos semanas, al vencer USD 6,000 millones en opciones, ¿qué viene a continuación?
  En un tuit de seguimiento tras la caída del jueves, Schultze-Kraft reafirmó que la caída "no era inesperada" y que, en su opinión, las perspectiva
  generale sigue siendo alcistas.
  "Estructuralmente, nada ha cambiado. Todavía no he visto un dato que apunte a la baja a largo plazo".
  Otra prueba de un posible cambio de rumbo a corto plazo puede encontrarse al observar el cambio en el suministro liquido de Bitcoin, que disminuyó en la
  mayor cantidad en más de 6 meses.
  Expand 1 lines ...
  Esto sugiere que un gran número de bitcoins han sido retirados del suministro circulante y han sido depositados en carteras de almacenamiento a largo
  plazo, ya que los alcistas se preparan para que el precio tienda a subir.
  Las altcoins se hunden
  La mayoría de las altcoins se vieron afectadas por la venta de Bitcoin, ya que los traders de todo el mercado cerraron sus posiciones en un intento de
  preservar sus ganancias.
- Desempeño diario del mercado de criptomonedas. Fuente: Coin360
  La única que destaca entre las altcoins es Aragon (ANT), cuyo reciente giro hacia DeFi y tokens no fungibles ha ayudado a desencadenar un repunte del
  50% que ha llevado su precio hasta los USD 13.56.
- El precio del token Theta cae un 25% de la noche a la mañana tras el retraso en el lanzamiento de la mainnet
  Holochain (HOT) y Balancer (BAL) también lograron ir en contra de la tendencia y consiguieron una ganancia positiva del 5.2% y 6.4% respectivamente.
  La capitalización total del mercado de criptomonedas actualmente es de USD 1.62 billones y la tasa de dominio de Bitcoin es del 59.4%.
- Los puntos de vista y opiniones expresados aquí son únicamente los del autor y no reflejan necesariamente los puntos de vista de Cointelegraph.com. Cada
  inversión y movimiento comercial implica riesgos, debes realizar tu propia investigación al tomar una decisión.
- El precio de Bitcoin rompe su mayor resistencia y un analista predice su "destino" en los $70,000
- El precio de Bitcoin se dirige al 'infinito', según el CEO de Kraken
- 6 tokens de DEX que han registrado un crecimiento impresionante en lo que va de 2021
- Fondo de pensiones de Nueva Zelanda asigna un 5% a Bitcoin
- #Aragon
- #Holo
- #Actualización de mercado
- #Bitcoin Options
- Las altcoins registran ganancias de tres dígitos mientras el precio de Bitcoin avanza hacia los 60,000 dólares
- Los alcistas de Bitcoin vuelven después de que BTC rebotara en un nivel de soporte clave
- Dogecoin (DOGE) y Maker (MKR) se disparan luego de que la capitalización de mercado de las altcoins superara el billón de dólares
- Binance ha nombrado un nuevo jefe para la "Gran China"
- El precio de Ethereum repunta frente a Bitcoin y los traders anticipan un fuerte rally para el par ETH/BTC
- Los flujos institucionales de XRP se disparan y los activos bajo gestión casi se duplican
- Bitpanda lanza servicio que permite invertir en acciones fraccionadas
- Criptan implementa el pago con criptomonedas en más de 2,000 hoteles
```

(c) Sample example difference between pipeline versions

FIGURE 3.2: High level statistics between two seperate pipelines and a sample example of the difference between two pipelines. First iteration (Figure 3.2a) generated a 7Mb dataset. After some careful tweaking, and some observed samples, we proposed a new pipeline in order to preserve previously wrongly removed data (Figure 3.2b) which generated a 134Mb dataset (x18).

3.4 Processing OSCAR

We chose to complement the data obtained at the end of the process described in the previous section with additional Common Crawl-based[5] data motivated by two main reasons. First, given the project's overall goal of providing a trained LLM as a research artifact comparable to previously released ones that have relied extensively on this source, we assessed that not including it would constitute too much of a departure and risk invalidating comparisons. Relatedly, recent work has put a strong emphasis on the quantity of data being a strong factor in a trained model's performance on evaluation tasks [Kaplan et al., 2020, Hoffmann et al., 2022a], and we were missing about one third of data in order to optimize our compute budget in this direction. With that in mind, we chose OSCAR version 21.09 [Ortiz Suárez et al., 2020], based on the Common Crawl snapshot of February 2021, to make up the remaining 38% of our final dataset.

However, crawled data suffers from several known issues. First, we wanted to only select documents written by humans for humans, and exclude machine-generated content e.g. search engine optimization (SEO). Crawled content also over-represents pornographic text across languages [Kreutzer et al., 2022], especially in the form of spam ads. Finally, it contains personal information that may constitute a privacy risk. The present section outlines our approach to mitigating those issues.

3.4.1 Data cleaning and filtering

Our first approach to addressing the above consists in defining quality indicators for web content. These can then be used to filter out specific pages by defining cutoff thresholds. We filtered out documents with:

- Too high **character repetition** or **word repetition** as a measure of repetitive content.
- Too high ratios of **special characters** to remove page code or crawling artifacts.
- Insufficient ratios of **closed class words** to filter out SEO pages.
- Too high ratios of **flagged words** to filter out pornographic spam. We asked contributors to tailor the word list in their language to this criterion (as opposed to generic terms related to sexuality) and to err on the side of high precision.
- Too high **perplexity** values to filter out non-natural language.
- Insufficient **number of words**, as LLM training requires extensive context sizes.

The languages that we eventually considered in OSCAR were the languages for which we were able to obtain hyperparameters and the cutoff values for each of these indicators by native speakers. Specifically, we considered Arabic, Basque, Bengali, Catalan, Chinese, English, French, Hindi, Indonesian, Portuguese, Spanish, Urdu, and Vietnamese. The code used for filtering OSCAR, along with the language-specific parameters and cutoff values, are publicly available. We then asked native speakers of each language to use our visualization tool[6] to establish the thresholds for each filter. The percentage of documents removed after applying all these filters is given in Table 3.2, and the percentage of documents discarded by each filter independently is given in 3.3.

[5]https://commoncrawl.org/

[6]https://hf.co/spaces/huggingface/text-data-filtering

AR	EU	BN	CA	ZH	EN	FR	HI	ID	PT	UR	VI	ES
20.3	5.2	48.8	21.1	23.1	17.2	17.0	25.7	10.4	12.6	15.8	21.3	16.9

TABLE 3.2: Percentage of documents removed by the filtering per language (ISO 639-1 code).

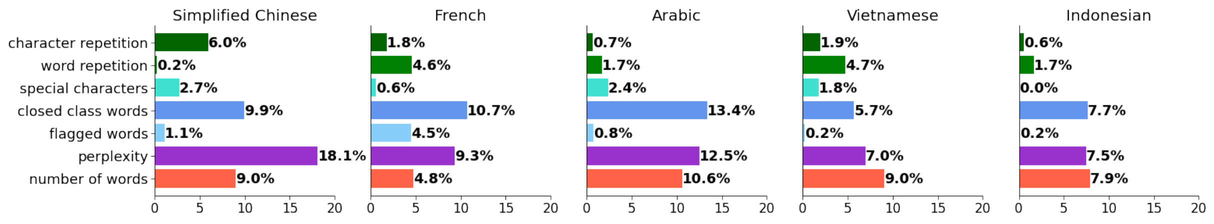


FIGURE 3.3: Percentage of documents discarded by each filter independently for 5 languages

3.4.2 Deduplication

Data deduplication has become a key tool for language model projects following research showing that it both improves performance on downstream tasks [Lee et al., 2022, Zhang et al., 2021] and decreases memorization of training data [Kandpal et al., 2022]. To remove near duplicate documents in OSCAR (which is already exact-deduplicated) we initially used SimHash [Charikar, 2002, Manku et al., 2007], a hashing function that associates to two similar texts hashes with a low Hamming distance, with 6-grams and a Hamming distance threshold of 4. About 0.7% of the documents on average (0.07% ~ 2.7%) were identified as near duplicates. However, because SimHash is essentially a bag-of-words algorithm, long documents are more likely to end up being similar to each other. In practice, we found false positives among long documents and decided not to discard documents in a same cluster of near-duplicates when they were longer than 6000 characters. Instead, we applied substring deduplication [Lee et al., 2022] based on Suffix Array [Manber and Myers, 1993] as a complementary method that clusters documents sharing a long substring, for documents with more than 6000 characters. We found on average 21.67% (10.61% ~ 32.30%) of the data (in bytes) being duplicated.

3.4.3 Personally identifiable information

We used a rule-based approach leveraging regular expressions. The elements redacted were instances of *KEY* (numeric & alphanumeric identifiers such as phone numbers, credit card numbers, hexadecimal hashes and the like, while skipping instances of years and simple numbers), *EMAIL* (email addresses), *USER* (a social media handle) and *IP_ADDRESS* (an IPv4 or IPv6 address).

3.5 A First look at ROOTS

The efforts described in the previous sections come together in an assemblage of 1.6 Terabytes of multilingual text. Figure 3.4 puts that number into context by comparing the sizes of corpora typically used to train large language models. Documentation of the individual components of the corpus can be found in an interactive dataset card deck. In this section, we take initial steps towards further understanding of the corpus through statistical analyses of the aggregated data.

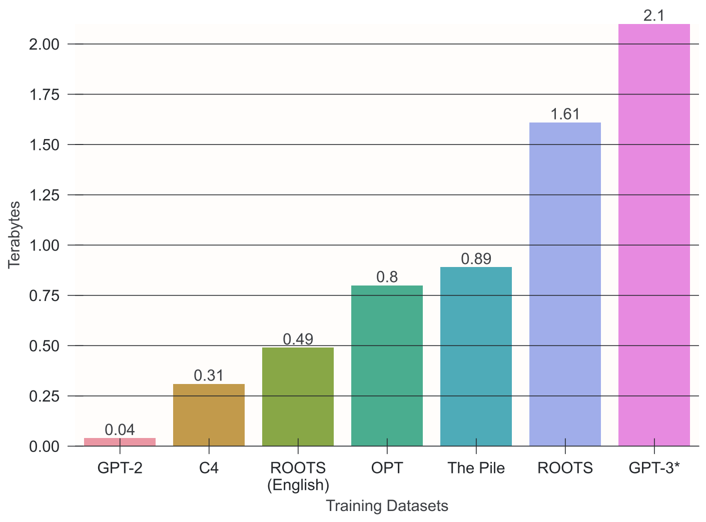


FIGURE 3.4: A raw size comparison to other corpora used to train large language models. The asterisk next to GPT-3 indicates the fact that the value in question is an estimate computed using the reported number of tokens and the average number of tokens per byte of text that the GPT-2 tokenizer produces on the `Pile-CC`, `Books3`, `OWT2`, and `Wiki-en` subsets of the Pile [Gao et al., 2020a]

3.5.1 Natural Languages

The constitution of the corpus reflects the crowdsourcing efforts that enabled its creation. It comprises of 46 natural languages spanning 3 macroareas and 9 language families: Afro-Asiatic, Austro-Asiatic, Austronesian, Basque, Dravidian, Indo-European, Mande, Niger-Congo, Sino-Tibetan. At 30.03%, English constitutes the largest part of the corpus, followed by Simplified Chinese (16.16%), French (12.9%), Spanish (10.85%), Portuguese (4.91%) and Arabic (4.6%). The exact composition with all languages is presented in presented in table 3.3.

In order for the trained model to have an opportunity to learn long dependencies, the training corpus needs to contain long sequences of coherent text. At the same time, the previous post-processing steps only reduced the size of the documents. The median size of a document in our corpus is 1,129 bytes. Figure 3.5 shows the distribution of document sizes by language. A more detailed breakdown of the size of corpus on an online interactive tool.[7].

The distributions of the filter values for the different filters introduced in Section 3.4.1 and languages, for the Catalogue, Pseudo-Crawl and OSCAR (filtered) data are available in an online demo[8]. Examples for English are shown in figure 3.6. The different distributions reflect the diversity of sourcing and filtering of our main components. A notable example is the flagged word filter, for which the distribution for OSCAR is skewed right compared to the catalogue even after filtering.

3.5.2 Programming Languages

As depicted in the waffle plot in figure 3.1, the code subset of the corpus spans 13 programming languages, with Java, PHP, and C++ accounting for more than half of all documents.

[7]https://hf.co/spaces/bigscience-data/document-sizes
[8]https://hf.co/spaces/bigscience-catalogue-lm-data/filter_values_distributions

Language	ISO-639-3	catalog-ref	Genus	Family	Macroarea	Size in Bytes
Akan	aka	ak	Kwa	Niger-Congo	Africa	70,1554
Arabic	arb	ar	Semitic	Afro-Asiatic	Eurasia	74,854,900,600
Assamese	asm	as	Indic	Indo-European	Eurasia	291,522,098
Bambara	bam	bm	Western Mande	Mande	Africa	391,747
Basque	eus	eu	Basque	Basque	Eurasia	2,360,470,848
Bengali	ben	bn	Indic	Indo-European	Eurasia	18,606,823,104
Catalan	cat	ca	Romance	Indo-European	Eurasia	17,792,493,289
Chi Chewa	nya	ny	Bantoid	Niger-Congo	Africa	1,187,405
Chi Shona	sna	sn	Bantoid	Niger-Congo	Africa	6,638,639
Chi Tumbuka	tum	tum	Bantoid	Niger-Congo	Africa	170,360
English	eng	en	Germanic	Indo-European	Eurasia	484,953,009,124
Fon	fon	fon	Kwa	Niger-Congo	Africa	2,478,546
French	fra	fr	Romance	Indo-European	Eurasia	208,242,620,434
Gujarati	guj	gu	Indic	Indo-European	Eurasia	1,199,986,460
Hindi	hin	hi	Indic	Indo-European	Eurasia	24,622,119,985
Igbo	ibo	ig	Igboid	Niger-Congo	Africa	14078,521
Indonesian	ind	id	Malayo-Sumbawan	Austronesian	Papunesia	19,972,325,222
Isi Zulu	zul	zu	Bantoid	Niger-Congo	Africa	8,511,561
Kannada	kan	kn	Southern Dravidian	Dravidian	Eurasia	2,098,453,560
Kikuyu	kik	ki	Bantoid	Niger-Congo	Africa	359,615
Kinyarwanda	kin	rw	Bantoid	Niger-Congo	Africa	40,428,299
Kirundi	run	rn	Bantoid	Niger-Congo	Africa	3,272,550
Lingala	lin	ln	Bantoid	Niger-Congo	Africa	1,650,804
Luganda	lug	lg	Bantoid	Niger-Congo	Africa	4,568,367
Malayalam	mal	ml	Southern Dravidian	Dravidian	Eurasia	3,662,571,498
Marathi	mar	mr	Indic	Indo-European	Eurasia	1,775,483,122
Nepali	nep	ne	Indic	Indo-European	Eurasia	2,551,307,393
Northern Sotho	nso	nso	Bantoid	Niger-Congo	Africa	1,764,506
Odia	ori	or	Indic	Indo-European	Eurasia	1,157,100,133
Portuguese	por	pt	Romance	Indo-European	Eurasia	79,277,543,375
Punjabi	pan	pa	Indic	Indo-European	Eurasia	1,572,109,752
Sesotho	sot	st	Bantoid	Niger-Congo	Africa	751,034
Setswana	tsn	tn	Bantoid	Niger-Congo	Africa	1,502,200
Simplified Chinese	—	zhs	Chinese	Sino-Tibetan	Eurasia	261,019,433,892
Spanish	spa	es	Romance	Indo-European	Eurasia	175,098,365,045
Swahili	swh	sw	Bantoid	Niger-Congo	Africa	236,482,543
Tamil	tam	ta	Southern Dravidian	Dravidian	Eurasia	7,989,206,220
Telugu	tel	te	South-Central Dravidian	Dravidian	Eurasia	2993407,159
Traditional Chinese	—	zht	Chinese	Sino-Tibetan	Eurasia	762,489,150
Twi	twi	tw	Kwa	Niger-Congo	Africa	1,265,041
Urdu	urd	ur	Indic	Indo-European	Eurasia	2,781,329,959
Vietnamese	vie	vi	Viet-Muong	Austro-Asiatic	Eurasia	43,709,279,959
Wolof	wol	wo	Wolof	Niger-Congo	Africa	3,606,973
Xhosa	xho	xh	Bantoid	Niger-Congo	Africa	14,304,074
Xitsonga	tso	ts	Bantoid	Niger-Congo	Africa	707,634
Yoruba	yor	yo	Defoid	Niger-Congo	Africa	89,695,835
Programming Languages	—	—	—	—		174,700,245,772

TABLE 3.3: Linguistic makeup of the corpus.

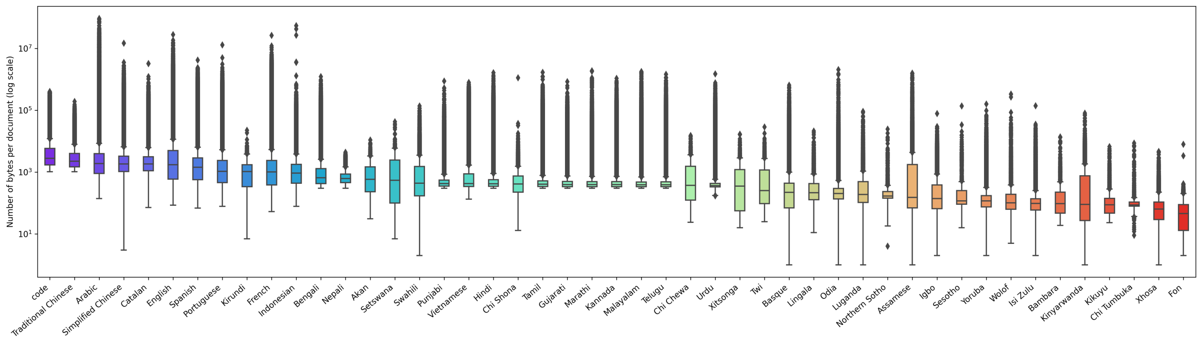


FIGURE 3.5: Size in bytes of every document in the corpus per language. The y-axis is in logarithmic scale. Box-and-whisker diagrams illustrate median, the first and third quartiles, whiskers drawn within the 1.5 IQR value and outliers

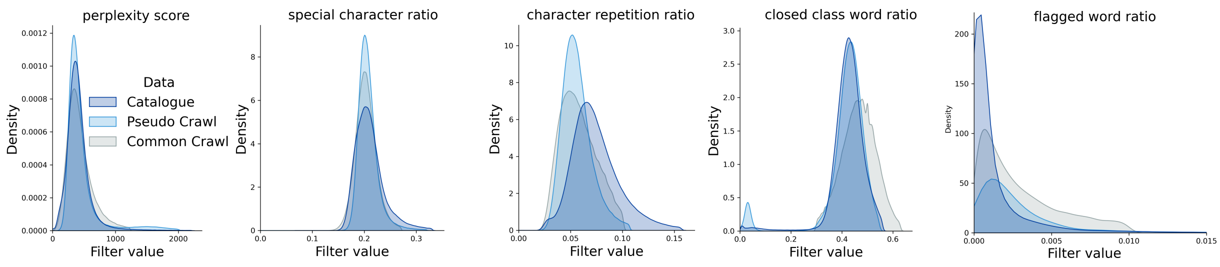


FIGURE 3.6: Some distributions of filter values for English. A filter value is the value that the filter gives to a document. These values are generally used to filter out documents that are too low or too high rated and also inform about the composition of the datasets.

Configuration and test files are abundant in most GitHub repositories but not as interesting for code modeling. To that end, we use a heuristic whose first step examines the first 5 lines of a file for the presence of keywords such as "configuration file" or "test file". Failing that, the second step is to see whether the occurrence of the literals `config` and `test` in a given file exceeds 5% of the total number of lines of that file. We find that 5.23% of the data consists of configuration files and 7.88% of test files.

Allamanis [2019] and Lopes et al. [2017] highlight the large fraction of near-duplicates present in code datasets and how they can inflate performance metrics. Exact match deduplication alone can miss a fair amount of near-duplicates. To detect them, we first compute the MinHash of all documents, then create a Locality Sensitive Hashing (LSH) index between files to find the duplicate clusters in linear time. We additionally evaluate the Jaccard similarities within duplicate clusters to remove some false positives. We find 10.9M duplicate files in the clusters and 4.1M unique files: almost 32% of the data consists of near-duplicates. Syntax checkers[9] are used to validate 500K samples of Python and PHP code. We find that only 1% of the Python data and 2% of the PHP files do not pass the syntax check.

3.5.3 Tokenizer analysis of the component datasets

A tokenizer trained on a dataset can be used as a proxy for its content [Gao et al., 2020a]. The relevant metric is the number of tokens produced for a byte of natural language. The more different the training

[9]`py_compile` for Python and the `-l` flag for PHP

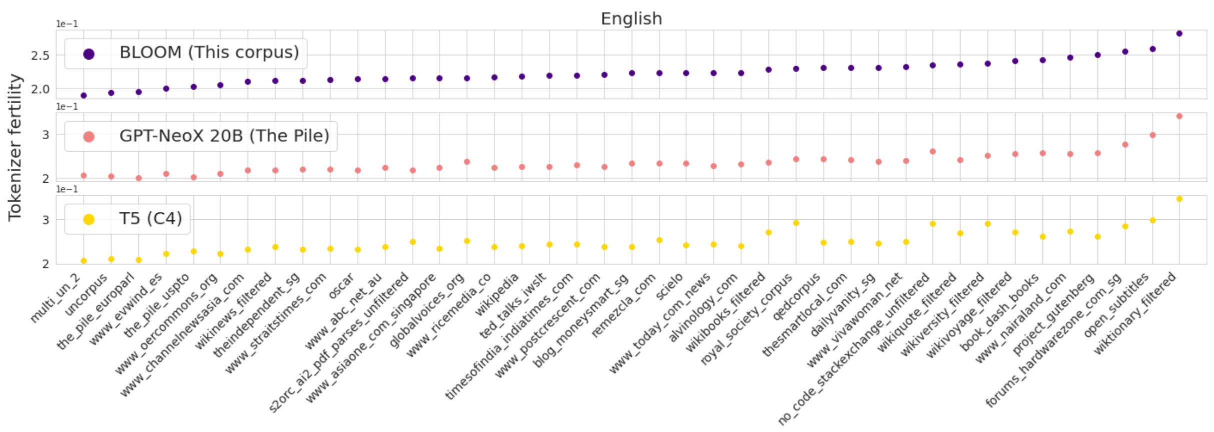


FIGURE 3.7: Tokens per byte for each English-language component for tokenizers trained on this corpus (BLOOM), the Pile (GPT-NeoX 20B) and C4 (T5). Lower values mean the component (X axis) is more similar in aggregate to the compared training corpus.

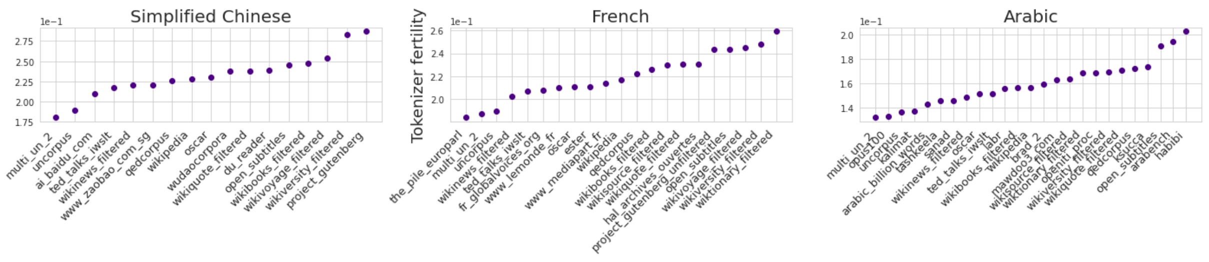


FIGURE 3.8: Tokens per byte for each French, Simplified Chinese, and Arabic component for tokenizers trained on this corpus. Lower values mean the component (X axis) is more similar in aggregate to the rest of the corpus.

corpus from the tokenized corpus, the more tokens will be produced as the tokenizer is forced to divide natural text in more numerous, more general, smaller tokens. This property has allowed us to spot errors associated with outlier values, such as incorrectly classified languages, or crawling error. In the following analysis, we use it in two ways: first, we can use tokenizers trained on different corpora to see how ours differs from them; and second, we can use a tokenizer trained on this corpus to assess which components are outliers. We exclude outliers smaller than 5 documents.

Figure 3.7 shows the tokens-per-byte measurement on English component datasets for the BLOOM tokenizer, trained on this corpus, the GPT-NeoX 20B tokenizer [Black et al., 2022], trained on the Pile, and the T5 tokenizer [Raffel et al., 2020], trained on C4. Those tokenizers may differ in algorithms and/or vocabulary size, but we won't be directly comparing them to each other.

The figure is ordered by BLOOM tokenizer token-per-byte values, which shows that the ordering is very similar for BLOOM and GPT-NeoX. However, it shows several bumps for T5: component datasets that are out of domain in C4 but not our corpus, for example technical and academic datasets such as `s2orc` or `royal_society_corpus`, domains absent from C4's Common Crawl-sourced data. Other such datasets include `global_voices`, which contains news about non-English-speaking regions including quotes in the original languages and `no_code_stackexchange`, which contains forums which, although in English, may be dedicated to technical matters, foreign languages, or very specific domains. Both are similar to our corpus but not to the Pile or C4.

Figure 3.8 additionally shows BLOOM fertilities for Simplified Chinese, French and Arabic components. Outlier, high-fertility components, e.g. datasets that differ from the rest of our corpus, tend to be the same for all languages. `project_gutenberg` contains old books with their original formatting (for example, "***********" to denote page ends). `wiktionary` contains definitions of words in foreign languages. `wikiversity` contains technical terms and LaTeX. `wikivoyage` contains tables formatted as text. Forums may contain the user and date information of the message, as well as internet slang or emoji. `arabench` is spoken Arabic, and `habibi` is classical Arabic with more diacritics than modern. We deem most of those deviations acceptable to represent the diversity of uses of text, which tokenizer analysis is able to surface from the rest of the dataset.

3.6 Conclusion

This chapter presented ROOTS, a massive multilingual corpus that was the result of an international collaboration with the goal of leveraging native speaker expertise for the processing in each language. We also released and documented the tooling developed throughout the project, in order to empower future language model training projects to make their own decisions with their data, and to spread best practices. For most languages in the dataset, ROOTS is still the largest publicly available language modeling corpus. Such large unstructured text corpora are a valuable resource for NLP communities as one of the two main inputs for large language model training, with computation power. In the next chapter, we will leverage ROOTS to train BLOOM, a massively multilingual large language model.

Chapter 4

Training an open-source massively multilingual language model

4.1 Introduction

As we have documented in chapter 2, large language models are the main engine of modern NLP, and those models are requiring increasing amounts of resources. Indeed, the empirical observation that a language model's performance increases predictably as the model is made larger [Hestness et al., 2017, Kaplan et al., 2020, Hoffmann et al., 2022b] has led to a trend of increasing scale [Zeng et al., 2021, Rae et al., 2021a, 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 are only affordable for well-resourced organizations. Furthermore, until recently, most were not publicly released. As a result, the majority of the research community is not involved in the development and investigation of LLMs. This has had concrete consequences; for example, most large language models are primarily trained on English-language text [1].

This chapter is dedicated to a multilingual language model we call BLOOM (the BigScience Large Open-science Open-access Multilingual Language Model) which was developed and released within the framework of BigScience, a collaboration of hundreds of researchers, with the goal of addressing these issues. From the dataset we built in chapter 4, we were able to train a 176 billion parameter model on 46 natural languages and 13 programming languages. At the time of writing, BLOOM was still the largest open-source model ever released, the only massively multilingual large language model, and the only publicly available language model over 20 billion parameters trained on publicly released data.

First, we document the entirety of the design of BLOOM, including architecture and modeling choices, engineering decisions, and some finetuned variants, with the goal of democratizing know-how around large language model training. Then, we report the results of our extensive evaluation suite, including performance on several tasks in monolingual and multilingual settings, the effects of finetuning, and some estimation of the bias exhibited by the model.

[1]with notable exceptions in Chinese and Korean, e.g. Wang et al., 2021, Zeng et al., 2021, Kim et al., 2021

4.2 BLOOM

4.2.1 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, which was presented in detail last chapter. A detailed itemized list of every language along with its linguistic genus, family and macroarea is presented in Table 4.1.

4.2.2 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. [2022b] 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.

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 dedicated 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. [2022b] 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., 2022b]. 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.

Language	ISO-639-3	catalog-ref	Genus	Family	Macroarea	Size in Bytes
Akan	aka	ak	Kwa	Niger-Congo	Africa	70,1554
Arabic	arb	ar	Semitic	Afro-Asiatic	Eurasia	74,854,900,600
Assamese	asm	as	Indic	Indo-European	Eurasia	291,522,098
Bambara	bam	bm	Western Mande	Mande	Africa	391,747
Basque	eus	eu	Basque	Basque	Eurasia	2,360,470,848
Bengali	ben	bn	Indic	Indo-European	Eurasia	18,606,823,104
Catalan	cat	ca	Romance	Indo-European	Eurasia	17,792,493,289
Chichewa	nya	ny	Bantoid	Niger-Congo	Africa	1,187,405
chiShona	sna	sn	Bantoid	Niger-Congo	Africa	6,638,639
Chitumbuka	tum	tum	Bantoid	Niger-Congo	Africa	170,360
English	eng	en	Germanic	Indo-European	Eurasia	484,953,009,124
Fon	fon	fon	Kwa	Niger-Congo	Africa	2,478,546
French	fra	fr	Romance	Indo-European	Eurasia	208,242,620,434
Gujarati	guj	gu	Indic	Indo-European	Eurasia	1,199,986,460
Hindi	hin	hi	Indic	Indo-European	Eurasia	24,622,119,985
Igbo	ibo	ig	Igboid	Niger-Congo	Africa	14078,521
Indonesian	ind	id	Malayo-Sumbawan	Austronesian	Papunesia	19,972,325,222
isiXhosa	xho	xh	Bantoid	Niger-Congo	Africa	14,304,074
isiZulu	zul	zu	Bantoid	Niger-Congo	Africa	8,511,561
Kannada	kan	kn	Southern Dravidian	Dravidian	Eurasia	2,098,453,560
Kikuyu	kik	ki	Bantoid	Niger-Congo	Africa	359,615
Kinyarwanda	kin	rw	Bantoid	Niger-Congo	Africa	40,428,299
Kirundi	run	rn	Bantoid	Niger-Congo	Africa	3,272,550
Lingala	lin	ln	Bantoid	Niger-Congo	Africa	1,650,804
Luganda	lug	lg	Bantoid	Niger-Congo	Africa	4,568,367
Malayalam	mal	ml	Southern Dravidian	Dravidian	Eurasia	3,662,571,498
Marathi	mar	mr	Indic	Indo-European	Eurasia	1,775,483,122
Nepali	nep	ne	Indic	Indo-European	Eurasia	2,551,307,393
Northern Sotho	nso	nso	Bantoid	Niger-Congo	Africa	1,764,506
Odia	ori	or	Indic	Indo-European	Eurasia	1,157,100,133
Portuguese	por	pt	Romance	Indo-European	Eurasia	79,277,543,375
Punjabi	pan	pa	Indic	Indo-European	Eurasia	1,572,109,752
Sesotho	sot	st	Bantoid	Niger-Congo	Africa	751,034
Setswana	tsn	tn	Bantoid	Niger-Congo	Africa	1,502,200
Simplified Chinese	—	zhs	Chinese	Sino-Tibetan	Eurasia	261,019,433,892
Spanish	spa	es	Romance	Indo-European	Eurasia	175,098,365,045
Swahili	swh	sw	Bantoid	Niger-Congo	Africa	236,482,543
Tamil	tam	ta	Southern Dravidian	Dravidian	Eurasia	7,989,206,220
Telugu	tel	te	South-Central Dravidian	Dravidian	Eurasia	2993407,159
Traditional Chinese	—	zht	Chinese	Sino-Tibetan	Eurasia	762,489,150
Twi	twi	tw	Kwa	Niger-Congo	Africa	1,265,041
Urdu	urd	ur	Indic	Indo-European	Eurasia	2,781,329,959
Vietnamese	vie	vi	Viet-Muong	Austro-Asiatic	Eurasia	43,709,279,959
Wolof	wol	wo	Wolof	Niger-Congo	Africa	3,606,973
Xitsonga	tso	ts	Bantoid	Niger-Congo	Africa	707,634
Yoruba	yor	yo	Defoid	Niger-Congo	Africa	89,695,835
Programming Languages	—	—	—	—		174,700,245,772

TABLE 4.1: Linguistic makeup of the ROOTS corpus.

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].

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., 2021a, 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].

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., 2022] 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. [2022b]. 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., 2022]. 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`[2] 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. [2022b], 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 `float16`, while the final training was in `bfloat16`. Since then, `float16` 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 `bfloat16` alleviates the need for the embedding LayerNorm.

We represent the full architecture of BLOOM in figure 4.1 for reference.

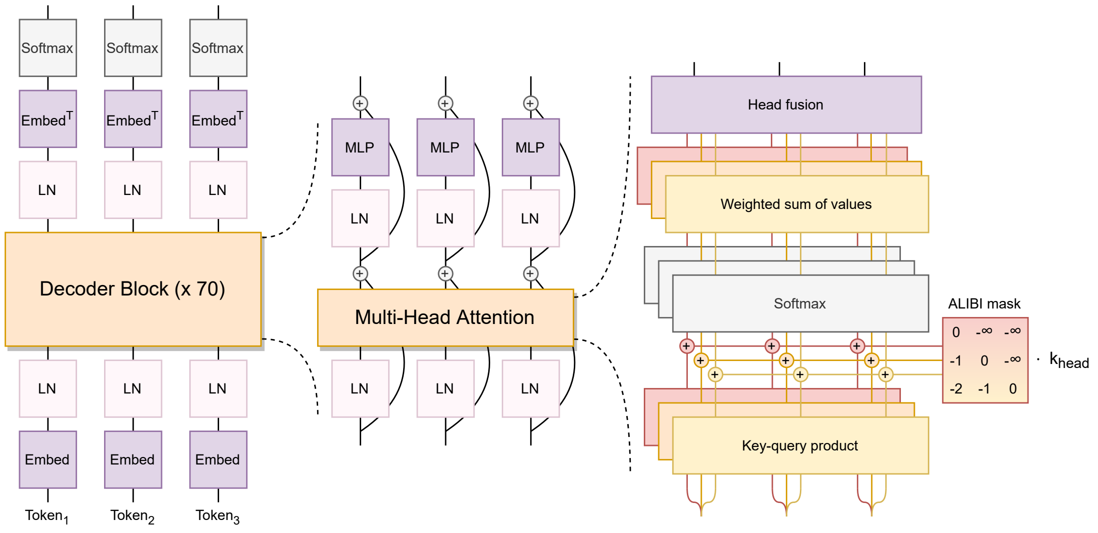


FIGURE 4.1: The BLOOM architecture. The k_{head} slope parameters for ALIBI are taken as $2^{\frac{-8i}{n}}$ with n the number of heads and $i \in 1, 2, ..., n$.

4.2.3 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

[2] `github.com/TimDettmers/bitsandbytes`

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.

Tokenizer	**fr**	**en**	**es**	**zh**	**hi**	**ar**
Monolingual	1.30	1.15	1.12	1.50	1.07	1.16
BLOOM	1.17 (-11%)	1.15 (+0%)	1.16 (+3%)	1.58 (+5%)	1.18 (+9%)	1.34 (+13%)

TABLE 4.2: 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 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., 2019c].

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 2^2 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|[.,!?…。，、।۔،])]+` ” [3] 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.

4.2.4 Engineering

Hardware

The model was trained on Jean Zay,[4] 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.

Framework

BLOOM was trained using Megatron-DeepSpeed[5] [Smith et al., 2022], a framework for large-scale distributed training. It consists of two parts: Megatron-LM[6] [Shoeybi et al., 2019] provides the Transformer implementation, tensor parallelism, and data loading primitives, whereas DeepSpeed[7] [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 4.2), a fusion of three complementary approaches to distributed training. These approaches are described below:

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

[3] `github.com/bigscience-workshop/bs-tokenizers`
[4] `idris.fr/eng/jean-zay/jean-zay-presentation-eng.html`
[5] `github.com/bigscience-workshop/Megatron-DeepSpeed`
[6] `github.com/NVIDIA/Megatron-LM`
[7] `github.com/microsoft/DeepSpeed`

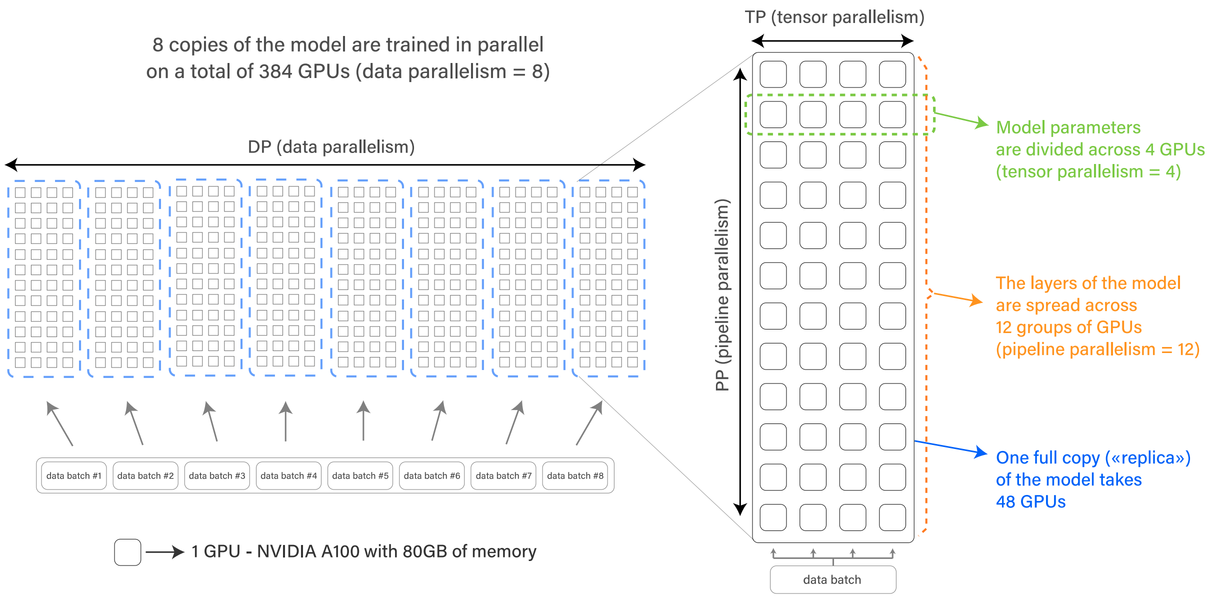

8 copies of the model are trained in parallel
on a total of 384 GPUs (data parallelism = 8)
DP (data parallelism)
data batch #1
data batch #2
data batch #3
data batch #4
data batch #5
data batch #6
data batch #7
data batch #8
1 GPU - NVIDIA A100 with 80GB of memory
TP (tensor parallelism)
PP (pipeline parallelism)
data batch
Model parameters are divided across 4 GPUs (tensor parallelism = 4)
The layers of the model are spread across 12 groups of GPUs (pipeline parallelism = 12)
One full copy («replica») of the model takes 48 GPUs

FIGURE 4.2: DP+PP+TP combination leads to 3D parallelism.

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) 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`).

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).

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.

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 checkpoints 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.[8]

4.2.5 Training

The BLOOM family We train six size variants of BLOOM with respective hyperparameters detailed in Table 4.3. Architecture and training hyperparameters come from our experimental results [Le Scao et al., 2022b] 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 β 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

[8] `github.com/bigscience-workshop/bigscience/blob/master/train/tr11-176B-ml/chronicles.md`

Hyperparameter (↓)	BLOOM-560M	BLOOM-1.1B	BLOOM-1.7B	BLOOM-3B	BLOOM-7.1B	BLOOM
			Architecture hyperparameters			
Parameters	559M	1,065M	1,722M	3,003M	7,069M	176,247M
Precision			`float16`			`bfloat16`
Layers	24	24	24	30	30	70
Hidden dim.	1024	1536	2048	2560	4096	14336
Attention heads	16	16	16	32	32	112
Vocab size			250,680			250,680
Sequence length			2048			2048
Activation			`GELU`			`GELU`
Position emb.			`Alibi`			`Alibi`
Tied emb.			`True`			`True`
			Pretraining hyperparameters			
Global Batch Size	256	256	512	512	512	2048
Learning rate	3.0e-4	2.5e-4	2e-4	1.6e-4	1.2e-4	6e-5
Total tokens			341B			366B
Warmup tokens			375M			375M
Decay tokens			410B			410B
Decay style			`cosine`			`cosine`
Min. learning rate			1e-5			6e-6
Adam (β_1, β_2)			(0.9, 0.95)			(0.9, 0.95)
Weight decay			1e-1			1e-1
Gradient clipping			1.0			1.0
			Multitask finetuning hyperparameters			
Global Batch Size	1024	1024	2048	2048	2048	2048
Learning rate	2.0e-5	2.0e-5	2.0e-5	2.0e-5	2.0e-5	2.0e-5
Total tokens			13B			13B
Warmup tokens			0			0
Decay style			`constant`			`constant`
Weight decay			1e-4			1e-4

TABLE 4.3: BLOOM & BLOOMZ Training Hyperparameters.

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., 2022b], 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.

4.2.6 Finetuned variants

Multitask Finetuning

Building on recent work on multitask finetuning [Sanh et al., 2022, Wei et al., 2021b, Wang et al., 2022a] we explore using *multilingual* multitask finetuning to improve the zero-shot performance of the BLOOM model. For this purpose, we use the xP3 corpus [Muennighoff et al., 2022b], resulting in a model we call BLOOMZ, further described in Muennighoff et al. [2022b]. BLOOMZ maintains the same architecture hyperparameters as BLOOM models. The finetuning hyperparameters are loosely based on T0 [Sanh et al., 2022] and FLAN [Wei et al., 2021b]. 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. Results on zero-shot performance are better across the board after multitask finetuning, as described in 4.3.7

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[9] geared towards multilingual information retrieval and SGPT-BLOOM-1.7B-nli[10] 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]. Results for some of those tasks are described in 4.3.8

4.2.7 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 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_2eq, 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).

[9]`hf.co/bigscience/sgpt-bloom-7b1-msmarco`
[10]`hf.co/bigscience-data/sgpt-bloom-1b7-nli`

Model name	Number of parameters	Power consumption	CO_2eq emissions
GPT-3	175B	1,287 MWh	*502 tons*
Gopher	280B	*1,066 MWh*	*352 tons*
OPT	175B	*324 MWh*	70 tons
BLOOM	176B	433 MWh	25 tons

TABLE 4.4: 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 4.4), 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_2eq/kWh, compared to 231 gCO_2eq/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_2eq 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].

4.2.8 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[11] [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)"[12] that the community has been adopting when releasing their models.[13] 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.

4.3 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 4.3.2, machine translation 4.3.3, summarization 4.3.4 - and languages in zero-shot and one-shot prompt-based settings, as well as after multitask finetuning (Section 4.3.7). We also perform code generation 4.3.5, and use BLOOM-derived text embeddings for representation tasks 4.3.8. Finally, we investigate biases shown by the model 4.3.9

4.3.1 Experimental Design

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

[11]hf.co/bigscience/bloom
[12]licenses.ai
[13]the-turing-way.netlify.app/reproducible-research/licensing/licensing-ml.html

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., 2022a]. 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 4.5 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 chapter 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.[14]

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

TABLE 4.5: 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").

Infrastructure

Our framework extends EleutherAI's Language Model Evaluation Harness [Gao et al., 2021] by integrating it with the `promptsource` [Bach et al., 2022a] library Bach et al. [2022a]. 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.

Datasets

SuperGLUE We use a subset of the SuperGLUE [Wang et al., 2019a] 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 magntiude 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

[14] `github.com/bigscience-workshop/promptsource/tree/eval-hackathon`

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.[15] 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],[16] 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., 2022]. 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.

Baseline Models

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

[15]BLEU+case:mixed+numrefs.1+smooth.exp+{13a,tok:spm-flores}+version:2.2.1

[16]For ROUGE, we used the Python implementation at `github.com/google-research/google-research/rouge`, commit `f935042`.

- mGPT [Shliazhko et al., 2022], GPT-style models trained on 60 languages from Wikipedia and Common Crawl
- GPT-Neo [Black et al., 2021], 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., 2020b]
- 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., 2022a]
- 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., 2020b]
- XGLM [Lin et al., 2021a], 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., 2022a], 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

4.3.2 SuperGLUE

Figure 4.3 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 4.3.7) 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.

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).[17] Figure 4.4 shows the accuracy of each prompt on each task across model scales.

[17]We do not evaluate OPT-66B because of the lack of a similarly-sized BLOOM model.

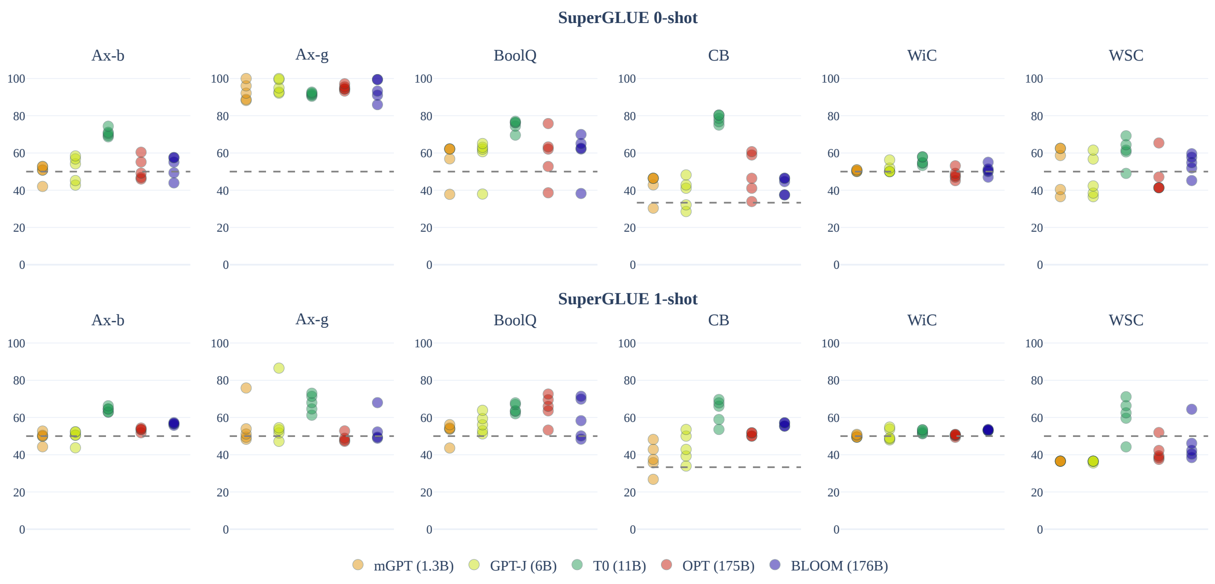


FIGURE 4.3: Performance of various LLMs on subset of tasks from SuperGLUE benchmark in zero- and one-shot prompt-based setting.

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.

4.3.3 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].

WMT

WMT results for BLOOM-176B in the zero-shot and 1-shot setting are given in Table 4.6. 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.

DiaBLa

Table 4.7 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

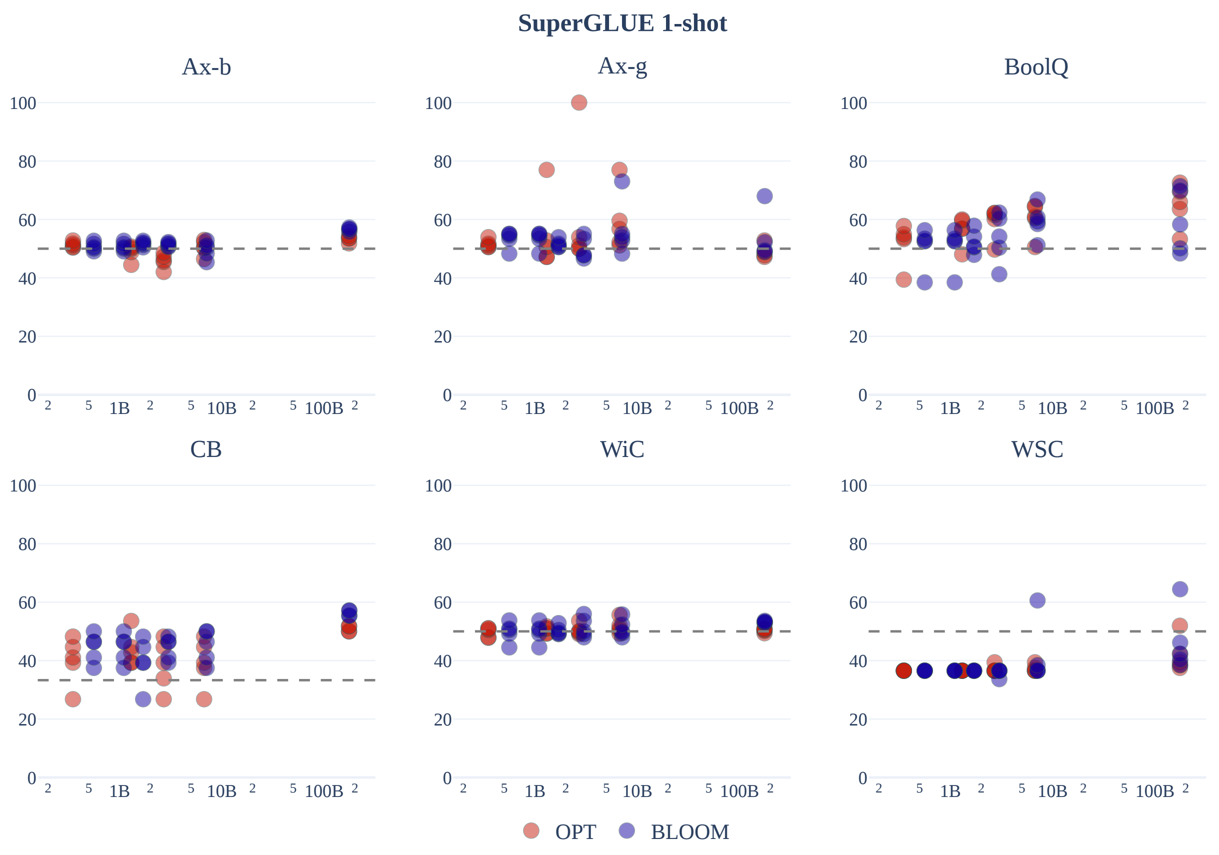


FIGURE 4.4: Comparison of the scaling of BLOOM versus OPT on each SuperGLUE one-shot task. Each point represents the average accuracy of a model within the BLOOM or OPT family of models on one of the five task prompts. The number of parameters on the x-axis is presented in log-scale.

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

TABLE 4.6: WMT'14 zero- and one-shot results (BLEU) for BLOOM-176B. The prompts used are described in Table 4.5.

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.[18] The automatic results are inconclusive, with little difference

[18]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

		en→fr		fr→en	
1-shot context	Truncate	BLEU	COMET	BLEU	COMET
Rand.	×	5.7	0.342	12.1	0.614
	✓	37.6	**0.634**	41.4	**0.757**
Prev.	×	6.1	0.328	12.3	0.617
	✓	**38.5**	0.614	**41.6**	0.751

TABLE 4.7: 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.

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.

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., 2021a]. The 1-shot example is randomly taken from the dev set. We separate out results for low-resource language pairs (Table 4.8a), between related languages of the Romance language family (Table 4.8b), high-resource language pairs (Table 4.8c) and high-to-mid-resource language pairs (Table 4.8d). 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., 2022a]. 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.

discarded.

Src↓	Trg→	en	bn	hi	sw	yo
en	BLOOM	–	24.6	27.2	20.5	2.6
	M2M	–	23.0	28.1	26.9	2.2
bn	BLOOM	29.9	–	16.3	–	–
	M2M	22.9	–	21.8	–	–
hi	BLOOM	35.1	23.8	–	–	–
	M2M	27.9	21.8	–	–	–
sw	BLOOM	37.4	–	–	–	1.3
	M2M	30.4	–	–	–	1.3
yo	BLOOM	4.1	–	–	0.9	–
	M2M	4.2	–	–	1.9	–

(A) Low-resource languages

Src↓	Trg→	ca	es	fr	gl	it	pt
ca	BLOOM	–	28.9	33.8	19.2	19.8	33.0
	M2M	–	25.2	35.1	33.4	25.5	35.2
es	BLOOM	31.2	–	24.8	23.3	16.5	29.1
	M2M	23.1	–	29.3	27.5	23.9	28.1
fr	BLOOM	37.2	27.5	–	24.9	24.0	38.9
	M2M	28.7	25.6	–	32.8	28.6	37.8
gl	BLOOM	37.5	27.1	33.8	–	18.3	32.2
	M2M	30.1	27.6	37.1	–	26.9	34.8
it	BLOOM	31.0	25.4	31.4	20.2	–	29.2
	M2M	25.2	29.2	34.4	29.2	–	31.5
pt	BLOOM	39.6	28.1	40.3	27.1	20.1	–
	M2M	30.7	26.9	40.2	33.8	28.1	–

(B) Romance languages

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

(C) High-resource language pairs.

Src ↓	Trg →	en	fr	hi	id	vi
en	BLOOM	–	45.0	27.2	39.0	28.5
	M2M	–	42.0	28.1	37.3	35.1
fr	BLOOM	45.6	–	18.5	31.4	32.8
	M2M	37.2	–	22.9	29.1	30.3
hi	BLOOM	35.1	27.6	–	–	–
	M2M	27.9	25.9	–	–	–
id	BLOOM	43.2	30.4	–	–	–
	M2M	33.7	30.8	–	–	–
vi	BLOOM	38.7	26.8	–	–	–
	M2M	29.5	25.8	–	–	–

(D) High→mid-resource language pairs.

TABLE 4.8: 1-shot MT results (spBLEU) on the Flores-101 devtest set.

4.3.4 Summarization

Figure 4.5 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 4.3.1, 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

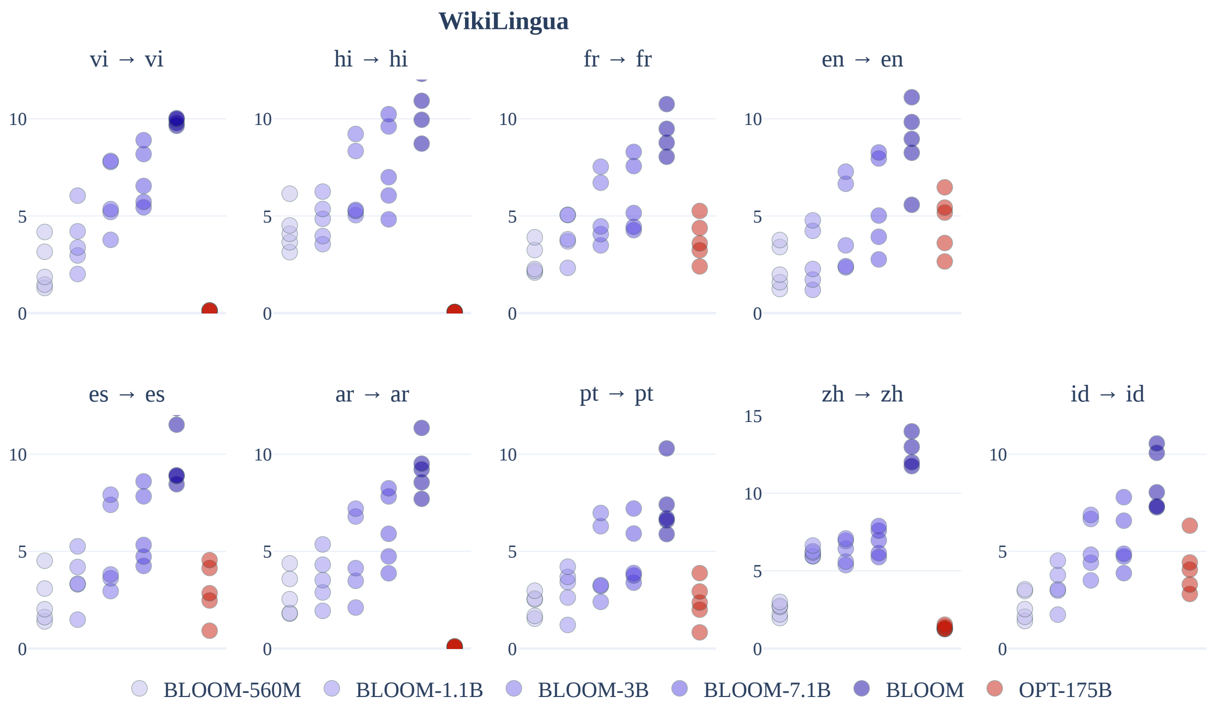


FIGURE 4.5: WikiLingua One-shot Results. Each plot represents a different language with per-prompt ROUGE-2 F-measure scores.

observe that in many cases, the ROUGE-2 score understates the quality of the summaries generated by the systems.

4.3.5 Code Generation

The BLOOM pretraining corpus, ROOTS, consists of around 11% of code. In Table 4.9, 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., 2020b]. 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].

4.3.6 HELM benchmark

For completeness, we reproduce here evaluations from the HELM benchmark [Liang et al., 2022a], 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

	PASS@k		
	$k=1$	$k=10$	$k=100$
GPT-NEO 1.3B	4.79%	7.47%	16.30%
GPT-NEO 2.7B	6.41%	11.27%	21.37%
GPT-J 6B	11.62%	15.74%	27.74%
GPT-NEOX 20B	15.4%	25.6%	41.2%
CODEX-300M	13.17%	20.37%	36.27%
CODEX-679M	16.22%	25.7%	40.95%
CODEX-2.5B	21.36%	35.42%	59.5%
CODEX-12B	28.81%	46.81%	72.31%
BLOOM-560M	0.82%	3.02%	5.91%
BLOOM-1.1B	2.48%	5.93%	9.62%
BLOOM-1.7B	4.03%	7.45%	12.75%
BLOOM-3B	6.48%	11.35%	20.43%
BLOOM-7.1B	7.73%	17.38%	29.47%
BLOOM	15.52%	32.20%	55.45%
BLOOMZ-560M	2.18 %	4.11%	9.00%
BLOOMZ-1.1B	2.63%	6.22%	11.68%
BLOOMZ-1.7B	4.38%	8.73%	16.09%
BLOOMZ-3B	6.29%	11.94%	19.06%
BLOOMZ-7.1B	8.06%	15.03%	27.49%
BLOOMZ	12.06%	26.53%	48.44%

TABLE 4.9: 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., 2021, Wang and Komatsuzaki, 2021, Black et al., 2022] are trained on a mix of code and text like BLOOM.

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.

4.3.7 Gains from multitask finetuning

We find that multitask finetuning causes zero-shot performance to significantly increase. In Figure 4.7, 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

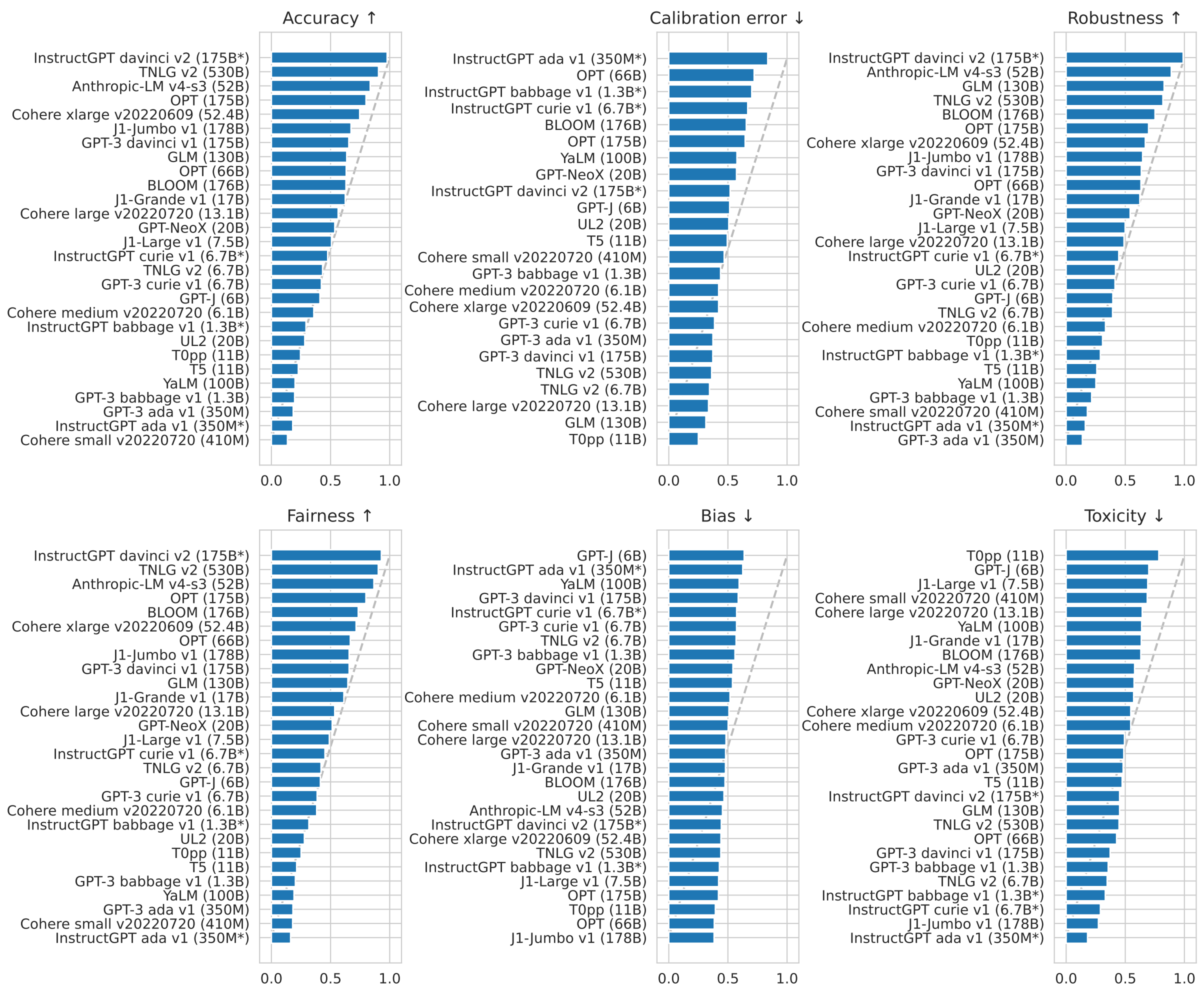


FIGURE 4.6: Results for a wide variety of language models on the 5-shot HELM benchmark. Taken from Liang et al. [2022a]

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].

4.3.8 Embeddings

In Section 4.2.5, we have outlined the contrastive finetuning procedure for creating SGPT-BLOOM text embedding models. In Table 4.10, 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[21] provides state-of-the-art performance on several classification and semantic

[21]hf.co/bigscience/sgpt-bloom-7b1-msmarco

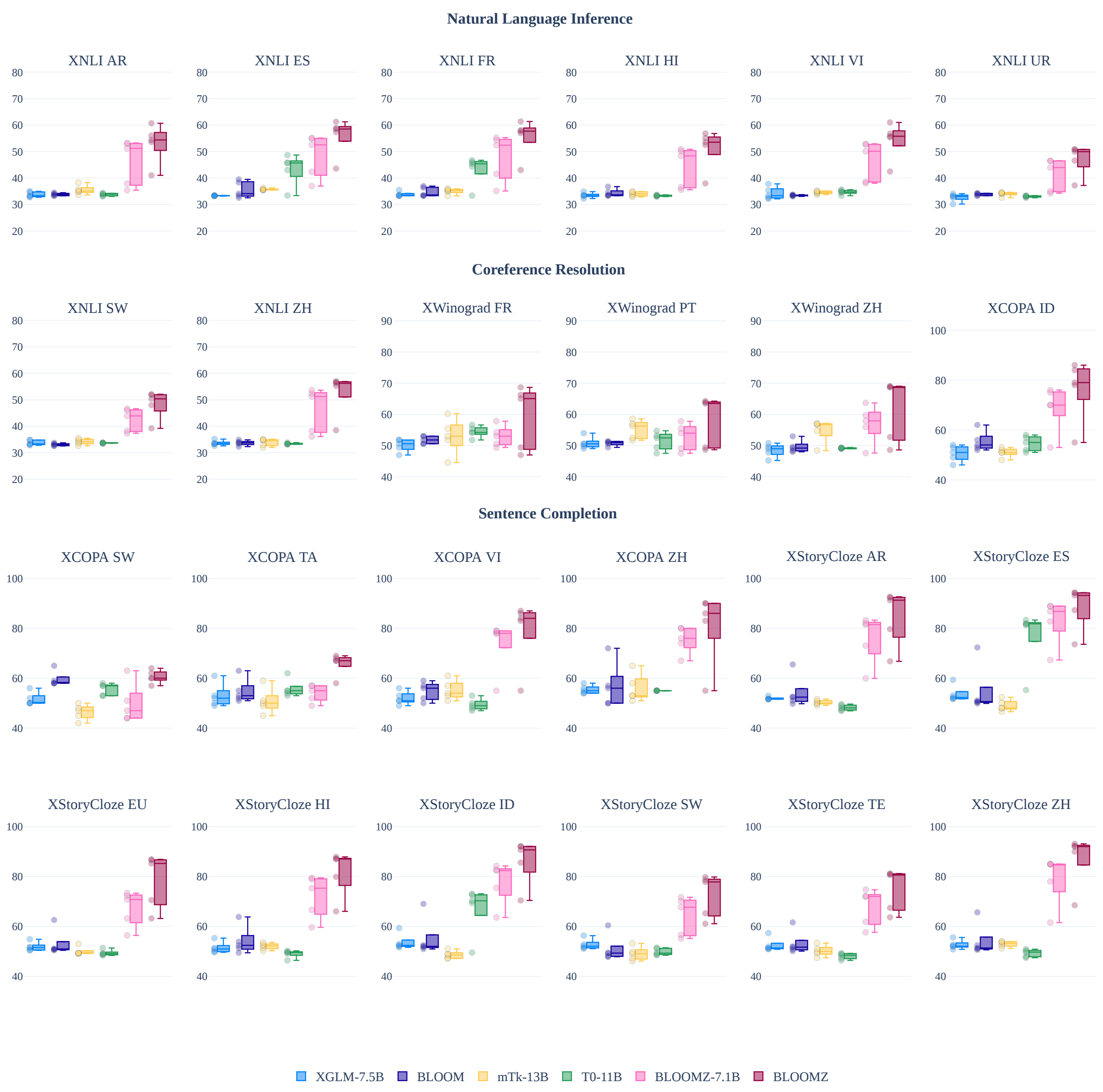


FIGURE 4.7: BLOOMZ zero-shot task generalization. Five untuned prompts are evaluated for each dataset and plotted. T0 is monolingual (English) while other models are multilingual. T0 performance may be hurt by its inability to tokenize some non-English texts.

textual similarity splits. However, with 7.1 billion parameters it is an order of magnitude larger than models like the displayed multilingual MiniLM[22] and MPNet[23]. SGPT-BLOOM-1.7B-nli[24] 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[25] is the largest model with 1.2 billion parameters. However, as an English-only model its performance on non-English languages is poor. The languages displayed are part of the BLOOM pretraining corpus. Performance on more

[22]hf.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
[23]hf.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
[24]hf.co/bigscience/sgpt-bloom-1b7-nli
[25]hf.co/sentence-transformers/sentence-t5-xl

	ST5-XL	LASER2	MiniLM-L12 [19]	MPNet[20]	LaBSE	SGPT-BLOOM-1.7B	SGPT-BLOOM-7.1B
Embedding classification performance on MASSIVE [FitzGerald et al., 2022] scored using accuracy							
Arabic (ar)	4.18	37.16	51.43	45.14	50.86	54.59	**59.25**
Bengali (bn)	2.60	42.51	48.79	35.34	58.22	57.76	**61.59**
English (en)	**72.09**	47.91	69.32	66.84	61.46	66.69	69.67
Spanish (es)	57.97	45.44	64.43	59.66	58.32	61.77	**66.35**
French (fr)	60.99	46.13	64.82	60.25	60.47	64.58	**66.95**
Hindi (hi)	3.02	40.20	62.77	58.37	59.40	60.74	**63.54**
Indonesian (id)	41.53	45.81	**65.43**	59.85	61.12	60.07	64.06
Kannada (kn)	2.79	4.32	50.63	40.98	**56.24**	48.56	53.54
Malayalam (ml)	2.98	41.33	54.34	42.41	57.91	55.10	**58.27**
Portuguese (pt)	57.95	48.55	64.89	61.27	60.16	62.52	**66.69**
Swahili (sw)	30.60	31.89	31.95	29.57	**51.62**	43.90	49.81
Tamil (ta)	1.79	29.63	50.17	36.77	55.04	52.66	**56.40**
Telugu (te)	2.26	36.03	52.82	40.72	**58.32**	49.32	54.71
Urdu (ur)	2.70	26.11	56.37	52.80	56.70	51.00	**56.75**
Vietnamese (vi)	21.47	44.33	59.68	56.61	56.67	59.85	**64.53**
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	63.02	58.53	**66.78**

TABLE 4.10: Performance of BLOOM models finetuned for sentence embeddings on classification and STS datasets from MTEB [Muennighoff et al., 2022b].

languages and datasets can be inspected on the MTEB leaderboard[26].

4.3.9 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

[26] `hf.co/spaces/mteb/leaderboard`

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 4.8 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., 2021] and GPT-FR [Simoulin and Crabbé, 2021] for English and French, respectively.

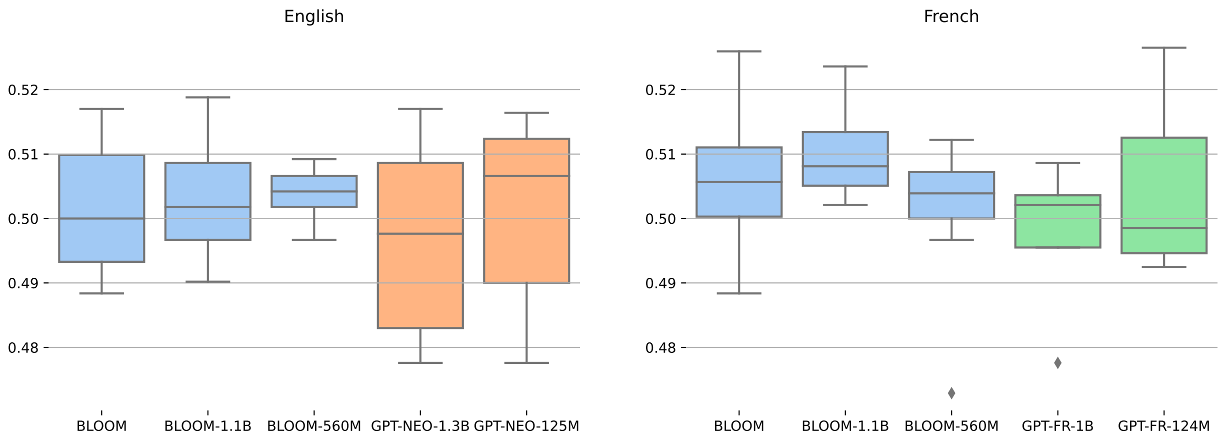


FIGURE 4.8: Overall accuracy of BLOOM on `crowS-Pairs` per prompt for English and French. Results on the two smallest BLOOM models and monolingual GPT models of comparable size are also shown.

Table 4.11 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, 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

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

TABLE 4.11: 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 *.

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].

4.4 Conclusion

In this chapter, we presented BLOOM, a 176B-parameter open-access multilingual language model. We also discussed evaluation results of BLOOM and other large language models, finding it has competitive performance that improves after multitask finetuning, and can serve as the base for a variety of applications in multilingual NLP, code generation, and information retrieval. At the time of release, BLOOM was the only model with zero-shot capabilities for many of its constituent languages. Such massively multilingual models, where English and other well-resourced corpora drive the bulk of the learning process and comparatively smaller amounts of data allow the model to generalize to other languages, are currently the best solution for access to strong pretrained models for communities with low amounts of linguistic data. For practitioners interested in creating monolingual — or multilingual with only a few languages, especially if not centered around English — models in languages with intermediary amounts of data, we dedicate the next two chapters to ways of making the most of the existing resources in data-constrained settings.

Chapter 5

Scaling laws under limited data

5.1 Introduction

The consensus in language model scaling laws has evolved quickly since the first large-scale studies began in Kaplan et al. [2020]. Recent work on compute-optimal language models [Hoffmann et al., 2022b] shows that many previously trained language models could have attained better performance for a given compute budget by training a smaller model on more data: their 70-billion parameter model [Hoffmann et al., 2022b], Chinchilla, outperforms the 280-billion parameter Gopher model [Rae et al., 2021a] by using a similar budget over four times more data. This is also the case for the previous chapter's work: at fixed compute, we know now that training a 57-billion parameter BLOOM over 1.13 trillion tokens would have yielded better performance. For most languages, available data is orders of magnitude smaller, meaning that LLMs in those languages are already data-constrained. Villalobos et al. [2022] estimate that even high-quality English language data will be exhausted by the year 2024 given those scaling laws and the trend of training ever-larger models hold [Villalobos et al., 2022, nostalgebraist, 2022]. This motivates the question of this chapter: *what should we do when we run out of data?*

To address it and help practitioners decide how to allocate compute, we investigate the scaling of large language models in a data-constrained regime. Multiple epochs are, of course, standard in machine learning generally; however, most prior large language models have been trained for a single epoch [Komatsuzaki, 2019, Brown et al., 2020] and some work explicitly advocates against reusing data [Hernandez et al., 2022]. In order to quantify the impact of data repetition, we run a battery of over 400 training experiments of varying data and compute constraints and record final test loss. We use these results to fit a new *data-constrained scaling law* that generalizes previous work [Hoffmann et al., 2022b] to the repeated data regime. Figure 5.1 summarizes our main results targeting the value of repeated data (*Return*) and optimal allocation of resources (*Allocation*). We find that, while training without repetition consistently yields the best validation loss per compute, those differences are insignificant among models trained for up to 4 epochs and do not lead to differences in downstream task performance. Additional epochs continue to be beneficial, but returns eventually diminish to zero. We also find that, in the data-constrained regime, the optimal allocation shifts slightly in favour of training over more data than the Chinchilla scaling law would predict. These findings suggest we can continue scaling training compute budgets further ahead in the future than previously anticipated.

Finally, given the challenges imposed by data constraints, we consider methods complementary to repeating for improving downstream accuracy without adding new natural language data. Experiments consider incorporating code tokens and relaxing data filtering. For code, English LLMs, such as PaLM [Chowdhery et al., 2022] or Gopher [Rae et al., 2021a], are trained on a small amount of code data alongside natural language data, though no benchmarking was reported to justify that decision. We investigate training LLMs on a mix of language data and Python data at 10 different mixing rates and find that mixing in code is able to provide a 2× increase in effective tokens even when evaluating only natural language tasks. For filtering, we revisit perplexity and deduplication filtering strategies on both noisy and clean datasets and find that data filtering is primarily effective for noisy datasets.

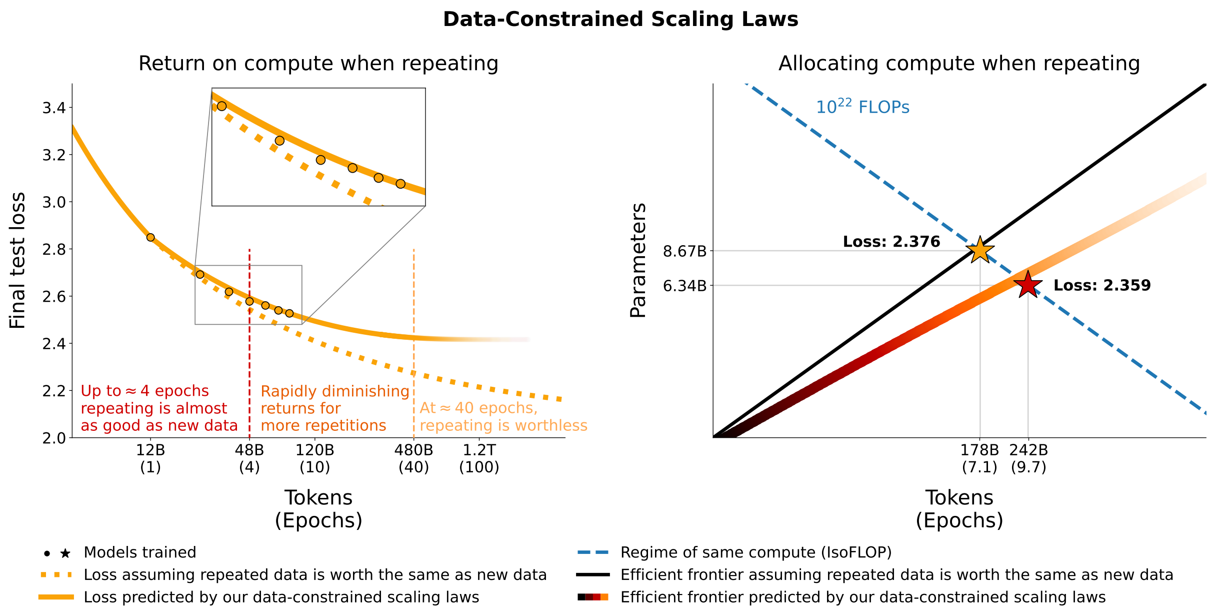


FIGURE 5.1: ***Return* and *Allocation* when repeating data.** *(Left):* Loss of LLMs (4.2B parameters) scaled on repeated data decays predictably (subsection 5.5.2). *(Right):* To maximize performance when repeating, our data-constrained scaling laws and empirical data suggest training smaller models for more epochs in contrast to what assuming Chinchilla scaling laws [Hoffmann et al., 2022b] hold for repeated data would predict (subsection 5.5.1).

5.2 Related Work

Large language models Scaling up transformer language models [Vaswani et al., 2017] across parameter count and training data has been shown to result in continuous performance gains [Chowdhery et al., 2022]. Starting with the 1.4 billion parameter GPT-2 model [Radford et al., 2019], a variety of scaled-up language models have been trained, commonly referred to as large language models (LLMs). They can be grouped into dense models [Brown et al., 2020, Khrushchev et al., 2022, Lieber et al., 2021, Rae et al., 2021a, Chung et al., 2022, Black et al., 2022, Zhang et al., 2022, Thoppilan et al., 2022, Su et al., 2022, Taylor et al., 2022, Zeng et al., 2022, Le Scao et al., 2022a, Li et al., 2023a] and sparse models [Fedus et al., 2022, Zeng et al., 2021, Du et al., 2022, Zoph et al., 2022] depending on whether each forward pass makes use of all parameters. These models are generally pre-trained to predict the next token in a sequence, which makes them applicable to various language tasks directly after pre-training [Brown et al., 2020, Wei et al., 2022, Kojima et al., 2022, Muennighoff, 2022, Srivastava et al., 2022] by reformulating said NLP tasks as context continuation tasks (see [Trinh and Le, 2018] for an earlier proposal on this topic). We focus on the most common scenario, where a dense transformer model is trained to do next-token prediction on a large corpus and evaluated directly after pre-training using held-out loss or zero- to few-shot prompting.

Scaling laws Prior work has estimated an optimal allocation of compute for the training of LLMs. Kaplan et al. [2020] suggested a 10× increase in compute should be allocated to a 5.5× increase in model size and a 1.8× increase in training tokens. This first scaling law has led to the creation of very large models trained on relatively little data, such as the 530 billion parameter MT-NLG model trained on 270 billion tokens [Smith et al., 2022]. More recent work [Hoffmann et al., 2022b], however, showed that model size and training data should rather be scaled in equal proportions. These findings called for a renewed focus on the scaling of pre-training data rather than scaling model size via complex parallelization strategies [Shoeybi et al., 2019, Rasley et al., 2020, Bian et al., 2021, Narayanan et al., 2021]. Up-sampling is often employed when pre-training data is partly limited, such as data from a high-quality domain like Wikipedia or text in a rare language for training multilingual LLMs [Lin et al., 2021b, Orlanski et al., 2023]. Hernandez et al. [2022] study up-sampling of data subsets and find that repeating only 0.1% of training data 100 times significantly degrades performance. In contrast, our work focuses on repeating the entire pre-training corpus for multiple epochs rather than up-sampling parts of it.

Alternative data strategies Large pre-training datasets are commonly filtered to remove undesired samples or reduce noise [Sorscher et al., 2022]. Perplexity-based filtering, whereby a trained model is used to filter out samples with high perplexity, has been found beneficial to reduce noise in web-crawled datasets [Wenzek et al., 2020]. Mixing of data is employed for the pre-training data of multilingual LLMs, where text data from different languages is combined [Conneau et al., 2020, Xue et al., 2020b, Soltan et al., 2022b, Muennighoff et al., 2022b]. However, both for code and natural language models, mixing different (programming) languages has been reported to under-perform monolingual models [Nijkamp et al., 2022, Virtanen et al., 2019]. Some work has investigated mixing code and natural language data for prediction tasks, such as summarizing code snippets [Iyer et al., 2016] or predicting function names [Allamanis et al., 2015]. Several pre-training datasets for LLMs include low amounts of code data [Gao et al., 2020b, Rae et al., 2021a, Le Scao et al., 2022a]. However, these past works generally do not provide any ablation on the drawbacks of including code or the benefits for natural language task performance. We perform a detailed benchmarking of mixing Python and natural language in LLM pre-training at 10 different mixing rates.

5.3 Data-constrained scaling laws

5.3.1 Background

Predicting the scaling behavior of large models is critical when deciding on training resources. Specifically, two questions are of interest: *(Allocation)* What is the optimal balance of resources? *(Return)* What is the expected value of additional resources? For scaling LLMs, the resource is compute (measured in

FLOPs), and it can be allocated to training a larger model or training for more steps.[1] The metric used to quantify progress is the model's loss on held-out data, i.e. the ability to predict the underlying data as measured in the model's cross-entropy [Alabdulmohsin et al., 2022, Hoffmann et al., 2022b]. We aim to minimize the loss (L) subject to a compute resource constraint (C) via optimal allocation to N and D as:

$$\operatorname*{argmin}_{N,D} L(N,D) \text{ s.t. } \text{FLOPs}(N,D) = C \tag{5.1}$$

Currently, there are established best practices for scaling LLMs. *Return* follows a power-law: loss scales as a power-law with the amount of compute used for training [Henighan et al., 2020, Kaplan et al., 2020, Bahri et al., 2021, Ghorbani et al., 2021, Bansal et al., 2022, Hernandez et al., 2021]. *Allocation* is balanced: resources are divided roughly equally between scaling of parameters and data [Hoffmann et al., 2022b]. These scaling laws were established empirically by training LLMs and carefully extrapolating behavior.

Chinchilla [Hoffmann et al., 2022b] uses three methods for making scaling predictions:

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

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

$$L(N,D) = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + E \tag{5.2}$$

Where $\{A, \alpha, B, \beta, E\}$ are learned variables fit using the training runs from the first two approaches [Hoffmann et al., 2022b]. Using these learned variables, they propose calculating the optimal allocation of compute (C) to N and D as follows:

$$\begin{gathered} N_{opt}(C) = G(C/6)^a \quad D_{opt}(C) = G^{-1}(C/6)^b \\ \text{where} \quad G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}} \quad a = \frac{\beta}{\alpha+\beta} \quad b = \frac{\alpha}{\alpha+\beta} \end{gathered} \tag{5.3}$$

These methods lead to the conclusion that $\alpha \approx \beta$ and hence N and D should be scaled proportionally for compute-optimal training. As loss can be an imperfect proxy for performance on natural language tasks [Xia et al., 2022, Shin et al., 2022, Tay et al., 2021], they also validate their conclusions on various downstream tasks.

5.3.2 Objectives

We are interested in scaling behavior in the data-constrained regime. Specifically, given a limited amount of unique data, what is the best *Allocation* of and *Return* for computational resources. Prior work [Kaplan et al., 2020, Hoffmann et al., 2022b] assumes that the necessary data to support scaling is

[1]In this chapter, we use [Kaplan et al., 2020]'s approximation for the compute cost: FLOPs$(N,D) \approx 6ND$, where N denotes the number of model parameters and D denotes the number of tokens processed.

unlimited. Our aim is therefore to introduce a modified version of Equation 5.2 that accounts for data constraints and fit the terms in the modified scaling law to data from a large body of experiments.

The primary method we consider is *repeating* data, i.e. allocating FLOPs to multiple epochs on the same data. Given a budget of unique data D_C, we split the Chinchilla total data term D into two parts: the number of unique tokens used, U_D, and the number of repetitions, R_D (i.e. epochs - 1). Given total training tokens D and data budget D_C these terms are simply computed as $U_D = \min\{D_C, D\}$ and $R_D = (D/U_D) - 1$. When training for a single epoch like done in prior scaling studies, $R_D = 0$. We are thus interested in minimizing Equation 5.1 with the additional constraint of a data budget D_C:

$$\underset{N,D}{\operatorname{argmin}}\, L(N,D) \text{ s.t. } \operatorname{FLOPs}(N,D) = C, U_D \leq D_C \tag{5.4}$$

Symmetrically, for mathematical convenience, we split the parameter term N into two parts: the base number of parameters needed to optimally fit the unique tokens U_N, and the number of times to “repeat” this initial allocation, R_N. We compute U_N by first rearranging Equation 5.3 to find the optimal compute budget for the unique tokens used (U_D). We input this value into the N_{opt} formula of Equation 5.3 to get $U_N = \min\{N_{opt}, N\}$. U_N thus corresponds to the compute-optimal number of parameters for U_D or less if $N < N_{opt}$. Once we have U_N, we compute the repeat value as $R_N = (N/U_N) - 1$.

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

- (*Fixed Unique Data*) In subsection 5.5.1 we fix the data constraint D_C and train models varying epochs and parameters. These experiments target *Allocation*, specifically tradeoff of D or N.
- (*Fixed FLOPs*) In subsection 5.5.2 we fix the computation available and vary D_C (and thus also U_D and U_N). These experiments target *Return*, i.e. how well does repeating scale compared to having more unique data.
- (*Parametric Fit*) We fit a formula introduced in subsection 5.3.3 on all our training runs and evaluate its predictive capability throughout subsection 5.5.1 and subsection 5.5.2.

Before discussing experimental results, we describe the parametric assumptions.

5.3.3 Adapting scaling theory to constrained data settings

Repeating data

Let N be the number of model parameters, D be the training tokens and U be the "unique" training tokens i.e. the size of the dataset to train on for one or more epochs. Chinchilla [Hoffmann et al., 2022b] only deals with non-repeated tokens. $D = U$ and we can start from Equation 5.2:

$$L(N,U) = \frac{A}{N^{\alpha}} + \frac{B}{U^{\beta}} + E \tag{5.5}$$

where E represents the irreducible loss. A, B, α and β are learned parameters.

We now want to generalize this expression to multiple epochs where tokens are repeated. We repeat the data R_D times, where $R_D = 0$ corresponds to the base case of a single epoch. We let D' be the "effective data size": the number of unique data needed to get the same value as repeating U unique tokens for R_D repeats. Hence, if $R_D = 0$, the effective data is the same as the total data processed. Intuitively, each time a sample is repeated, it is worth less as the model has already learned some of its information. Assume that each time a model trains on a token, it learns a $1-\delta$ fraction of the information in it for some constant $0 \leq \delta \leq 1$. (Thus, if $\delta = 0$ repeated tokens are as good as new ones, and if $\delta = 1$, repeated tokens are worth nothing.) This is an *exponential decay* assumption, where the value of a data token processed loses roughly $(1-\delta)$ fraction of its value per repetition, where δ is a learned constant. As we would like to sum up the value of all repetitions, we temporarily assume an integral number of repeats and express it as a geometric series:

$$D' = U + (1-\delta)U + (1-\delta)^2 U + \cdots + (1-\delta)^{R_D} U \tag{5.6}$$

We know that the sum S of a geometric series with a common ratio r is:

$$S = \frac{a(1-r^n)}{1-r} \tag{5.7}$$

where a is the first term and n the number of terms in the series. As $r = (1-\delta)$ and $a = (1-\delta)U$:

$$D' = U + U\sum_{k=1}^{R_D}(1-\delta)^k = U + (1-\delta)U\frac{(1-(1-\delta)^{R_D})}{\delta} \tag{5.8}$$

Note that Equation 5.8 can also be used with a non-integer number of repetitions. We can directly use Equation 5.8 as our effective data and learn δ but for convenience and interpretability, we redefine it in terms of the number of epochs beyond which repeating does not help. Note that as more data is repeated, the right-hand side tends to $\frac{(1-\delta)U}{\delta}$, as $\lim_{R_D\to\infty}(1-(1-\delta)^{R_D}) = 1$. Let $R_D^* = \frac{1-\delta}{\delta}$, hence D' "plateaus" at $U + R_D^* U$ as R_D goes to infinity.

If we assume δ to be small, $1-\delta$ tends to one and we can approximate $1/R_D^* = \frac{\delta}{1-\delta} \approx \delta$.

Next, define e^x in terms of its Taylor series expansion:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \approx 1 + x \tag{5.9}$$

If x is small later terms become increasingly small, thus $e^x \approx 1 + x$. As we have assumed δ to be small, let $x = -\delta$, which yields

$$(1+x) = (1-\delta) \approx e^{-\delta} \approx e^{-1/R_D^*} \tag{5.10}$$

Now inserting $(1-\delta)/\delta = R_D^*$ and $(1-\delta)^{R_D} = e^{(-1/R_D^*)^{R_D}}$ into Equation 5.8 we get our final equation representing the *effective data*:

$$D' = U + U \cdot R_D^* \cdot (1 - e^{-R_D/R_D^*}) \tag{5.11}$$

where U and R_D are given while R_D^* is a learned constant. If no repeats are done, the second part of the sum is zero and the term simplifies to the single-epoch scaling laws from Equation 5.5. While $R_D \ll R_D^*$, the second term is approximated as $U \cdot R_D$ and for $R_D \gg R_D^*$, it plateaus at $U \cdot R_D^*$. Hence R_D^* corresponds to the upper limit of the usefulness of repetition.

Repeating parameters

Similarly, we consider repeating parameters. Symmetric to seeing the same data, excess parameters learn the same features and do not add any value in the extreme. For the Chinchilla equation (Equation 5.5) increasing parameters from 1 billion to 10 billion yields the same absolute decrease in loss regardless of whether the dataset is a single token or 1 billion tokens. However, intuition suggests that in the first case, adding parameters should not decrease loss at all, as the additional 9 billion parameters cannot possibly learn anything from the single token that the first 1 billion parameters have not already learned. Thus, to allow excess parameters to decay to adding nothing, we also replace N with a symmetric version of Equation 5.11 yielding our final equation:

$$L(U_N, U_D, R_N, R_D) = \frac{A}{(U_N + U_N R_N^*(1 - e^{\frac{-R_N}{R_N^*}}))^\alpha} + \frac{B}{(U_D + U_D R_D^*(1 - e^{\frac{-R_D}{R_D^*}}))^\beta} + E \tag{5.12}$$

We define U_N, as the number of "unique" parameters that provide an optimal fit for U_D. Additional parameters decay with a symmetric version of the expression for repeated data. R_N is the number that the "unique" parameters are repeated i.e. $R_N = \max\{(N/U_N) - 1, 0\}$. If $R_N^* = \infty$, additional parameters do not decay at all and $(U_N + U_N R_N^*(1 - e^{\frac{-R_N}{R_N^*}}))$ reduces to N. We compute U_N from U_D by setting $D_{opt} = U_D$ and rearranging Equation 5.3 to map from D_{opt} to N_{opt}. U_N is then $\min\{N_{opt}, N\}$. This is equivalent to the following:

$$U_N = \min\{((U_D \cdot G)^{\beta/\alpha}) \cdot G, N\} \quad \text{where} \quad G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}} \tag{5.13}$$

Equation 5.12 is a generalization of Equation 5.5: It provides the same estimates for optimal model and data size in the single epoch case, but allows for decay in the value of parameters and tokens, thus generalizing to training for multiple epochs and with excess parameters.

5.4 Experimental Setting

For all experiments, we train transformer language models with the GPT-2 architecture and tokenizer [Radford et al., 2019]. Models have up to 8.7 billion parameters and are trained for up to 900 billion total tokens. Following [Hoffmann et al., 2022b] we use cosine learning rate schedules that decay 10× over the course of training for each model (different schedules led to different estimates in [Kaplan et al., 2020]). Unlike [Kaplan et al., 2020], we do not use early stopping to also explore the extent of overfitting when repeating. Other hyperparameters are based on prior work [Rae et al., 2021a, Hoffmann et al., 2022b]. Models are trained on subsets of C4 [Raffel et al., 2020]. The data constraints are carefully defined to ensure maximal overlap as shown in Figure 5.2. Unlike [Hernandez et al., 2022], we always repeat the entire available data rather than subsets of it. Data is shuffled after each epoch. As

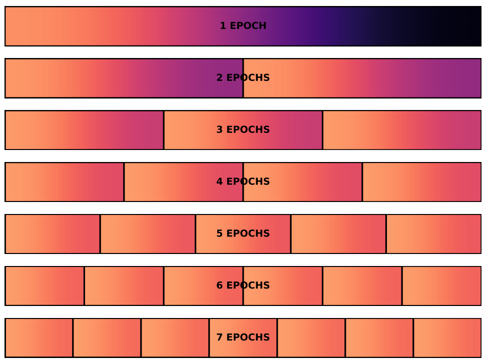


FIGURE 5.2: **Dataset setup.** Training runs with different epochs reuse subsets of the same data to ensure different training data is not a confounding factor.

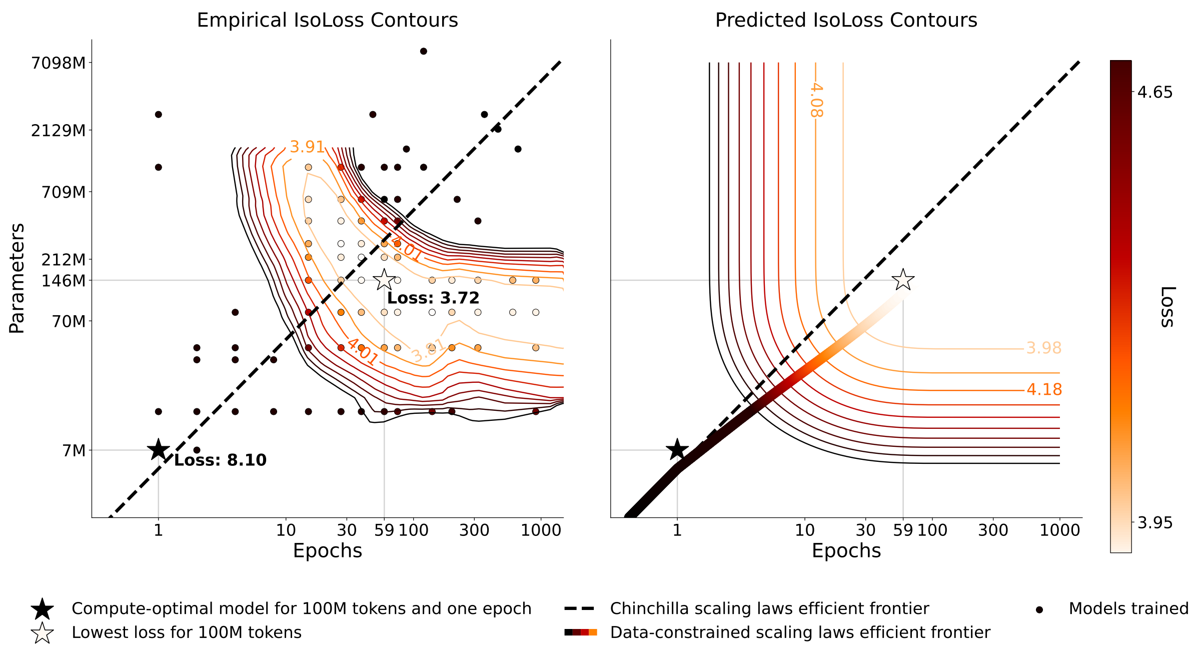


FIGURE 5.3: **IsoLoss contours for 100 million unique tokens.** *(Left):* 93 models trained with varying parameters and epochs on a fixed dataset. Contours show an interpolation of results with the same final test loss. *(Right):* Comparison with the loss predictions from our proposed scaling laws for the same budget of 100 million unique tokens and the predicted efficient frontier. The diminishing returns from training on repeated data can be seen in the increase in distance of the contour curves.

repeating data can result in extreme overfitting, we report loss on a held-out test set. This contrasts training loss used in [Hoffmann et al., 2022b], but should not alter our findings as the held-out data stems from the same underlying dataset.

5.5 Results

5.5.1 Resource Allocation for Data-Constrained Scaling

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

Figure 5.3 (left) shows the main results for scaling with 100M unique tokens[2] For 100M tokens, the corresponding one-epoch compute-optimal model according to scaling laws from [Hoffmann et al., 2022b] has U_N of approximately 7M parameter. Results show that more than a 50% reduction in loss can be attained by training for several epochs ($R_D > 0$) and increasing model size beyond what would be compute optimal for 100M tokens ($R_N > 0$). We find the best loss to be at around 20-60$\times$ more parameters and epochs, which corresponds to spending around 7000$\times$ more FLOPs. These results

[2]Although small, for example, this is the order of magnitude of a realistic data constraint reflecting data available after filtering the OSCAR dataset [Ortiz Suárez et al., 2019] for Basque, Punjabi, or Slovenian.

suggest that one-epoch models significantly under-utilize their training data and more signal can be extracted by repeating data and adding parameters at the cost of sub-optimal compute utilization.

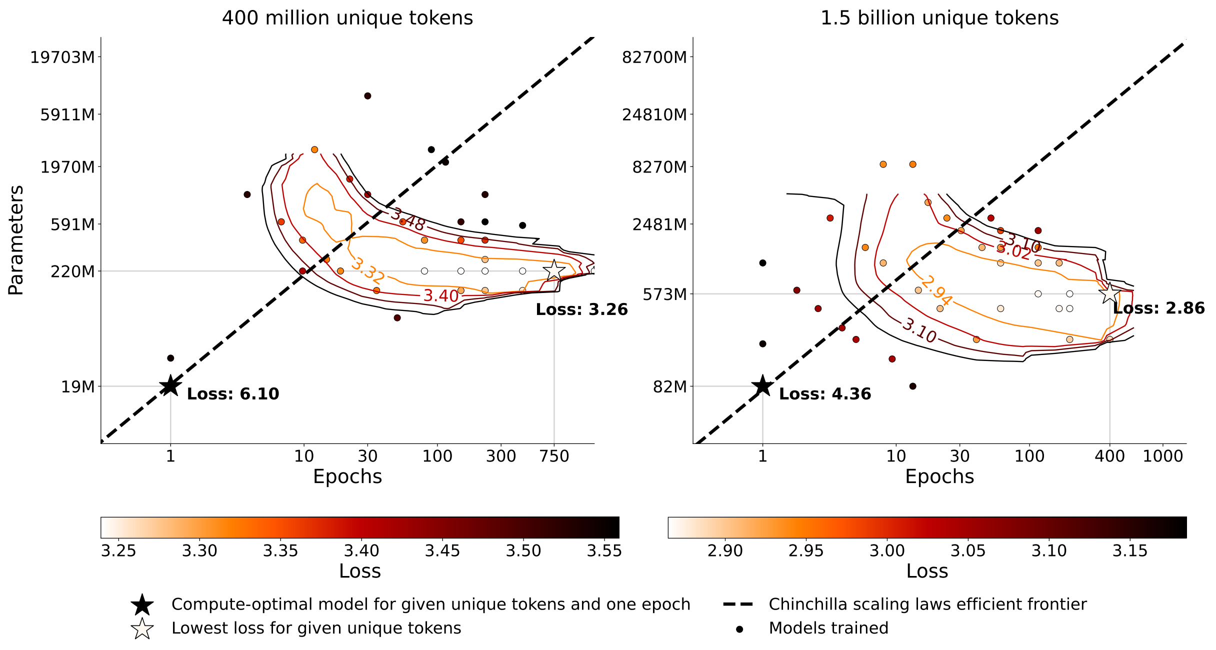
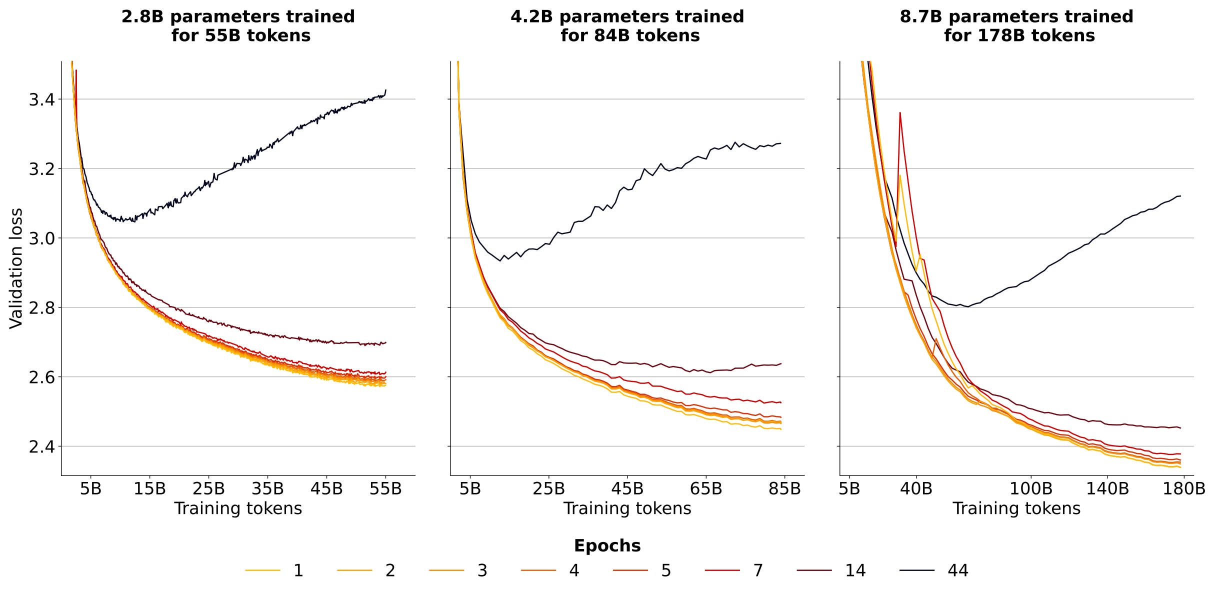


FIGURE 5.4: **Empirical isoLoss curves for 400 million and 1.5 billion unique tokens.** 34 models trained on 400 million unique tokens and 37 models trained on 1.5 billion unique tokens with varying parameters and epochs.

Figure 5.3 (right) shows the predicted contours created by fitting our data-constrained scaling laws on 182 training runs. In the single-epoch case ($R_D = 0$) with near compute-optimal parameters ($R_N = 0$) our scaling equation (subsection 5.3.3) reduces to the Chinchilla equation. In this case, both formulas predict the optimal allocation of compute to parameters and data to be the same, resulting in overlapping efficient frontiers. As data is repeated for more than a single epoch, our fit predicts that excess parameters decay faster in value than repeated data ($R_N^* < R_D^*$). As a result, the data-constrained efficient frontier suggests allocating most additional compute to more epochs rather than more parameters. This contrasts the Chinchilla scaling laws [Hoffmann et al., 2022b], which suggest equally scaling both. However, note that they do not repeat the entire training data and their parametric fit explicitly relies on the assumption that models are trained for a single epoch only. Thus, there is no guarantee that their scaling predictions hold for repeated data.

Figure 5.4 contains additional empirical isoLoss contours for 400 million and 1.5 billion unique tokens. Results show that like in Figure 5.3 significantly lower loss can be achieved by increasing parameters and epochs beyond what is compute-optimal at a single epoch. The lowest loss is also achieved by allocating more extra compute to repeating data rather than to adding parameters.

For all three data budgets, our results suggest that *Allocation* is optimized by scaling epochs faster than additional parameters. We confirm this at scale by training the data-constrained compute-optimal model for 9.3×10^{21} FLOPs and 25 billion unique tokens as suggested by our efficient frontier. Despite having 27% less parameters, this model achieves better loss than the model suggested by the Chinchilla scaling laws (Figure 5.1, right). Similarly, the 120 billion parameter Galactica model trained on repeated



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

FIGURE 5.5: **Validation Loss for Different Data Constraints (IsoFLOP).** Each curve represents the same number of FLOPs spent on an equal size model. Colors represent different numbers of epochs due to repeating because of data constraints. Parameters and training tokens are set to match the single-epoch compute-optimal configurations for the given FLOPs. Models trained on data that is repeated for multiple epochs have consistently worse loss and diverge if too many epochs are used.

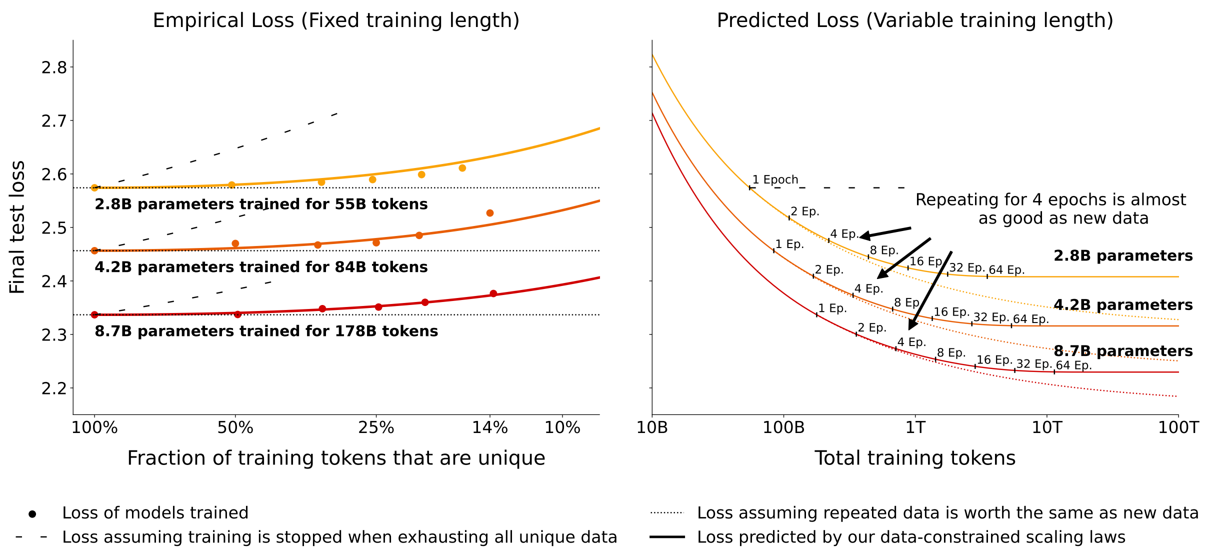


FIGURE 5.6: **Empirical and Extrapolated loss with constrained data.** *(Left):* Loss as a function of repeated tokens for three different training budgets each with fixed number of parameters. Loss curves predicted by our data-constrained scaling laws are shifted to exactly match the loss at 100% unique data. Return on FLOPs decays with repeated data in a regular pattern. *(Right):* Extrapolating from the proposed data-constrained scaling law shows that at small numbers epochs are benign, but at large number of epochs loss stops improving.

data should have been significantly smaller according to data-constrained scaling laws. An additional benefit of using a smaller model is cheaper inference, though adding parameters can make it easier to parallelize training across GPUs.

Adding parameters and epochs causes the loss to decrease and eventually increase again, suggesting that too much compute can hurt performance. Results from [Kaplan et al., 2020] also show that loss can increase when too many parameters are used, even with early stopping. However, we expect that appropriate regularization (such as simply removing all excess parameters as an extreme case) could prevent this behavior. Thus, our formula presented in subsection 5.3.3 and its predicted isoLoss contours in Figure 5.3 do not model the possibility that excess epochs or parameters could hurt performance.

5.5.2 Resource Return for Data-Constrained Scaling

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

Figure 5.5 shows the configurations and validation curves for models trained on the same number of total tokens. Conforming to intuition and prior work on deduplication [Lee et al., 2022], repeated data is worth less, thus models trained on less unique data (and, correspondingly, more epochs) have consistently higher loss. However, the loss difference for a few epochs is negligible. For example, the $N = 8.7$ billion parameter model trained for four epochs ($D_C = 44$ billion unique tokens) finishes training with only 0.5% higher validation loss than the single-epoch model ($D_C = 178$ billion unique tokens).

In Figure 5.6 (left), we compare the final test loss of each model to predictions from our parametric fit. The data-constrained scaling laws can accurately measure the decay in the value of repeated data as seen by the proximity of empirical results (dots) and parametric fit (lines). We note however that it significantly underestimates the final test loss of failing models where loss increases midway through training, such as models trained for 44 epochs (not depicted).

In Figure 5.6 (right), we extrapolate the three budgets by further scaling compute while keeping the data constraints (D_C) at 55B, 84B, and 178B tokens, respectively. The parameter R_D^* introduced in subsection 5.3.3 represents roughly the "half-life" of epochs: specifically the point where repeated tokens have lost $\frac{1}{e}$ of their value. Through our fitting in subsection 5.5.3, we found $R_D^* \approx 15$, corresponding to 15 repetitions (or 16 epochs). Graphically, this can be seen by the stark diminishing returns in the proximity of the 16-epoch marker and the flattening out soon after.

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

5.5.3 Parametric Fit

Using a methodology similar to [Hoffmann et al., 2022b], R_N^* and R_D^* can be fit on empirical measurements, which yields data-driven estimates. We use the LBFGS algorithm to find local minima of the objective above, started on a grid of initialization given by: $R_N^* \in \{0., 4., \ldots, 20.\}$ and $R_D^* \in \{0., 4., \ldots, 20.\}$. We fit on 182 samples with parameters varying from 7 million up to 9 billion and epochs ranging from 1 to 500. We removed runs where performance decreased with extra resources, as our formulas do not allow for excess parameters or excess epochs to negatively impact performance. We assume excess parameters or epochs only cause performance to plateau but never to worsen. However, it is difficult to identify all samples where excess parameters or epochs hurt, as for some data budgets we only train a single model, thus we do not know if the loss of that model is already in the range where it starts to increase again. Further, there are samples where loss initially increases and then decreases as a function of epochs[3], which further contributes to noise in the fitting.

Nevertheless, we are able to get a fairly stable fit resulting in $R_N^* = 5.309743$ and $R_D^* = 15.387756$. Since $R_D^* > R_N^*$, excess parameters decay faster. Hence, the data-constrained efficient frontiers in Figures 5.1,5.3 suggest scaling compute allocated to epochs faster than to parameters. This value of R_D^* yields $\delta \approx 6 * 10^{-2}$ (0.19 for R_N^*), which respects the assumption that δ is small. Inserting these learned parameters and simplifying Equation 5.13 yields the precise formulation we use to predict loss (L) given unique tokens (U_N), parameter repetitions (R_N) and data repetitions (R_D):

$$L(U_D, R_N, R_D) = \frac{521}{(U_N + 5.3 \cdot U_N(1 - e^{\frac{-R_N}{5.3}}))^{0.35}} + \frac{1488}{(U_D + 15.4 \cdot U_D(1 - e^{\frac{-R_D}{15.4}}))^{0.35}} + 1.87 \tag{5.14}$$

$$\text{with } U_N = U_D \cdot 0.051$$

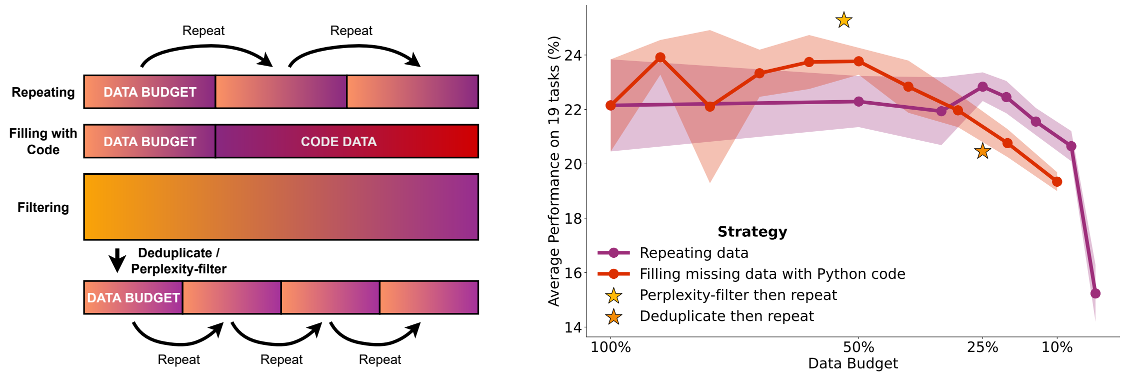


FIGURE 5.7: **Strategies for data-constrained settings and their downstream performance.** *(Left):* Schematic showing alternative data use strategies of code filling and filtering. *(Right):* $N = 4.2$ billion parameter models trained for a total of $D = 84$ billion tokens with varying budgets D_C. For repeating and filling with code, five models with different seeds are trained for each dot and the standard deviation is visualized as the shaded area.

5.6 Complementary Strategies for Obtaining Additional Data

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

Figure 5.7 (left) illustrates the strategies: **(a) Code augmentation:** We use Python code from The Stack [Kocetkov et al., 2022] to make up for missing natural language data. The combined dataset consisting of code and natural language samples is shuffled randomly. **(b) Adapting filtering:** We investigate the performance impact of deduplication and perplexity filtering, two common filtering steps that can severely limit available data. Removing such filtering steps can free up additional training data.

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

When comparing across data strategies, loss ceases to be a good evaluation metric as the models are trained on different data distributions. We thus evaluate models on 19 natural language tasks with zero to five in-context few-shot exemplars [Brown et al., 2020] producing 114 scores per model. As our evaluation tasks cover different metrics and random baselines, we re-scale all scores to be in the same range to better reflect performance ranges before averaging. For each dataset, a maximum of 3000 samples are evaluated with 0,1,2,3,4 and 5 few-shots [Brown et al., 2020] to produce six scores which are then averaged. Prompts are sourced from GPT-3 [Brown et al., 2020] and PromptSource [Bach et al., 2022b].

[3]A phenomenon usually referred to as double descent [Nakkiran et al., 2021, Hernandez et al., 2022]

In Figure 5.7 (right) we compare the downstream performance of all strategies. For repeating data, differences in downstream performance are insignificant for up to around 4 epochs (25% budget) and then start dropping, which aligns with our results on test loss in subsection 5.5.2. Filling up to 50% of data with code (42 billion tokens) also shows no deterioration. Beyond that, performance decreases quickly on natural language tasks. However, adding more code data may benefit non-natural language tasks, which are not considered in the benchmarking. Two of the tasks benchmarked, WebNLG [Castro Ferreira et al., 2020, Gehrmann et al., 2021], a generation task, and bAbI [Weston et al., 2015, Liang et al., 2022b], a reasoning task, see jumps in performance as soon as code is added, possibly due to code enabling models to learn long-range state-tracking capabilities beneficial for these tasks.

Of the filtering approaches, we find perplexity-filtering to be effective, while deduplication does not help. Prior work found deduplication was able to improve perplexity [Lee et al., 2022]; however, it did not evaluate on downstream tasks. Deduplication may have value not captured in our benchmark, such as reducing memorization [Kandpal et al., 2022, Hernandez et al., 2022, Carlini et al., 2022, Biderman et al., 2023]. Overall, in a data-constrained regime, we recommend reserving filtering for noisy datasets and using both code augmentation and repeating to increase data tokens. For example, first doubling the available data by adding code and then repeating the new dataset for four epochs results in 8× more training tokens that are expected to be just as good as having had 8× more unique data from the start.

5.7 Conclusion

This chapter studies data-constrained scaling, focusing on the optimal use of computational resources when unique data is limited. We propose an extension to the Chinchilla scaling laws that takes into account the decay in value of repeated data, and we fit this function using a large set of controlled experiments. We find that despite recommendations of earlier work, training large language models for multiple epochs by repeating data is beneficial, with diminishing returns only apparent after over four epochs. We also consider complementary approaches to continue scaling models, and find that code gives the ability to scale an additional 2×. We believe that those findings will enable further scaling of language models to unlock new capabilities with current data, which is especially pressing in non-English languages. For now, we've only considered pretraining; in the next chapter, we will concern ourselves with data requirements at finetuning time.

Chapter 6

Large language models and prompting help with downstream task data requirements

6.1 Introduction

Now that we have spent a few chapters building pretrained models, it is time to use them to actually solve NLP tasks. For this, the older paradigm was to replace the token classification head of the base model by an explicit classifier head, then to finetune [Devlin et al., 2018, Howard and Ruder, 2018]. With the progress of pretraining, it has become possible to use them directly as zero-shot predictors through autoregressive text generation Radford et al. [2019] or completion of a cloze task Trinh and Le [2018]. The former is generally used in low-resource languages, where pretrained models are not as strong, or if a single task has to be applied a large number of times and one wishes to use smaller models. The latter is generally used when there is a large number of different tasks to solve and good pretrained models are available, or if training data for the task is scarce.

One argument made for the second method is that it allows us to pick custom *prompts* for each task [McCann et al., 2018]. While this approach was originally pioneered by zero-shot classification [Puri and Catanzaro, 2019] or priming [Brown et al., 2020], prompts have since then also be used in the first method to provide extra task information to the classifier, especially in the low-data regime [Schick and Schütze, 2020a,b]. It is natural then to ask how it impacts the sample efficiency of the model, or more directly, *how many data points is a prompt worth?* As with many low-data and pretraining-based problems, this question is complicated by the finetuning setup, training procedure, and prompts themselves. We attempt to isolate these variables through diverse prompts, multiple runs, and best practices in low-training data finetuning. We introduce a metric, the *average data advantage*, for quantifying the impact of a prompt in practice.

In this chapter, we use this method to unify the two paradigms, with zero-shot classification a data-less extreme case case of prompt-based finetuning. We find that the impact of task-targeted prompting can nicely be quantified in terms of direct training data, and that it varies over the nature of different tasks. On MNLI [Williams et al., 2018], we find that using a prompt contributes approximately 3500 data points. On SuperGLUE, it adds approximately 280 data points on RTE [Dagan et al., 2006] and up to 750 on BoolQ [Clark et al., 2019]. In low- to medium-data settings, such as rare tasks or lower-resource languages, this advantage can be a real contribution to training a model.

6.2 Related Work

Prompting has been used both for zero-shot and finetuning based methods. Zero-shot approaches attempt to answer a task with a prompt without finetuning through generation [Radford et al., 2019]. GPT3 [Brown et al., 2020] extends this approach to a supervised priming method by taking in training data as priming at inference time, so it can attend to them while answering. T5 [Raffel et al., 2020] and other sequence-to-sequence pretrained models use standard word-based finetuning with a marker prompt to answer classification tasks with strong empirical success. Our setting differs in that we are interested in using task-based prompts and finetuning, in-between the T5 and GPT2 setting.

Our setting most closely resembles PET [Schick and Schütze, 2020a,b], which claims that task-specific prompting helps transfer learning, especially in the low-data regime. However, in order to reach the best possible results on SuperGLUE, PET introduces several other extensions: semi-supervision via additional pseudo-labeled data, ensembling models trained with several different prompts, and finally distilling the ensemble into a linear classifier rather than a language model. Our aim is to isolate the specific contributions of prompting within supervised finetuning.

Finally, recent papers have experimented with discovering prompts through automated processes tailored to the language model [Jiang et al., 2020, Schick et al., 2020]. We limit ourselves to human-written prompts, as we are interested into whether prompting itself specifically adds information to the supervised task. It is an interesting question as to whether automatic prompts can have this same impact (relative to the training data they require).

6.3 Heads and Prompts

Consider two transfer learning settings for text classification: *head-based*, where a generic head layer takes in pretrained representations to predict an output class; *prompt-based*, where a task-specific pattern string is designed to coax the model into producing a textual output corresponding to a given class. Both can be utilized for finetuning with supervised training data, but prompts further allow the user to customize patterns to help the model.

For the *prompt* model we follow the notation from PET [Schick and Schütze, 2020a] and decompose a prompt into a *pattern* and a *verbalizer*. The *pattern* turns the input text into a cloze task, i.e. a sequence with a masked token or tokens that need to be filled. Let us use as example an excerpt from SuperGLUE task BoolQ [Clark et al., 2019], in which the model must answer yes-or-no binary questions. In order to let a language model answer the question in *italics*, our pattern is in **bold** [Schick and Schütze, 2020b]:

> "Posthumous marriage – Posthumous marriage (or necrogamy) is a marriage in which one of the participating members is deceased. It is legal in France and similar forms are practiced in Sudan and China. Since World War I, France has had hundreds of requests each year, of which many have been accepted. **Based on the previous passage,** ***can u marry a dead person in france ?*** **<MASK>**"

The masked word prediction is mapped to a *verbalizer* which produces a class. (here "Yes": True. "No": False[1]). Several *pattern-verbalizer pairs* (*PVPs*) could be used for a single task, differing either

[1]The correct answer here is, of course, *yes*. Originated in 1803 as Napoleon rose to power, this practice was

through the pattern, the verbalizer, or both. Finetuning is done by training the model to produce the correct verbalization. The loss is the cross-entropy loss between the correct answer and the distribution of probabilities amongst the tokens in the verbalizer. We re-use pattern choices from Schick and Schütze [2020b].

6.4 Experimental Setting

We run all experiments with the same pretrained checkpoint, *roberta-large* (355M parameters) from RoBERTa [Liu et al., 2019], which we load from the *transformers* [Wolf et al., 2020] library.[2] In line with previous observations [McCoy et al., 2019, Dodge et al., 2020, Lee et al., 2020], head-based finetuning performance varies considerably. We follow recommendations of Mosbach et al. [2020] and Zhang et al. [2020b] to train at a low learning rate (10^{-5}) for a large number of steps (always at least 250, possibly for over 100 epochs).

We perform our evaluation on SuperGLUE and MNLI [Williams et al., 2018]. These datasets comprise a variety of tasks, all in English, including entailment (MNLI, RTE [Dagan et al., 2006], CB [De Marneffe et al., 2019]), multiple choice question answering (BoolQ [Clark et al., 2019], MultiRC [Khashabi et al., 2018]), and common-sense reasoning (WSC [Levesque et al., 2012], COPA [Roemmele et al., 2011], WiC [Pilehvar and Camacho-Collados, 2018]). We do not include ReCoRD [Zhang et al., 2018] in our comparisons as there is no head model to compare with, since it is already a cloze task. Data sizes range from 250 data points for CB to 392,702 for MNLI. As test data is not publicly available for SuperGLUE tasks, we set aside part of training (from 50 for CB, COPA and MultiRC to 500 for BoolQ) to use for development, and evaluate on their original validation sets. For MNLI, we use the available matched validation and test sets.

We compare models across a scale of available data, starting with 10 data points and increasing exponentially (as high-data performance tends to saturate) to the full dataset. For example, for MultiRC, which has 969 data points initially, we start by reserving 50 data points for development. This leaves us with 919 training points, and we train models with 10, 15, 20, 32, 50, 70, 100, 150, 200, 320, 500, 750, and 919 training points. We run every experiment 4 times in order to reduce variance, for a total of 1892 training runs across all tasks. At every point, we report the best performance that has been achieved at that amount of data or lower. This means that if the maximum performance over random seeds is smaller than a maximum previously attained with less data points, we use the previous value. Using this or the maximum at each data point gives broadly equivalent results; using the mean, however, can make results vary significantly, as the distribution of outcomes is heavily left-skewed, or even bimodal, with poor-performance outliers.

6.5 Results

Figure 6.1 shows the main results comparing head- and prompt-based finetuning with the best-performing pattern on that task. Prompting enjoys a substantial advantage on every task, except for

mainly to the benefit of war widows.

[2]After experimenting with RoBERTa, AlBERT [Lan et al., 2019] and BERT [Devlin et al., 2018], we found *roberta-large* to have the most consistent performance.

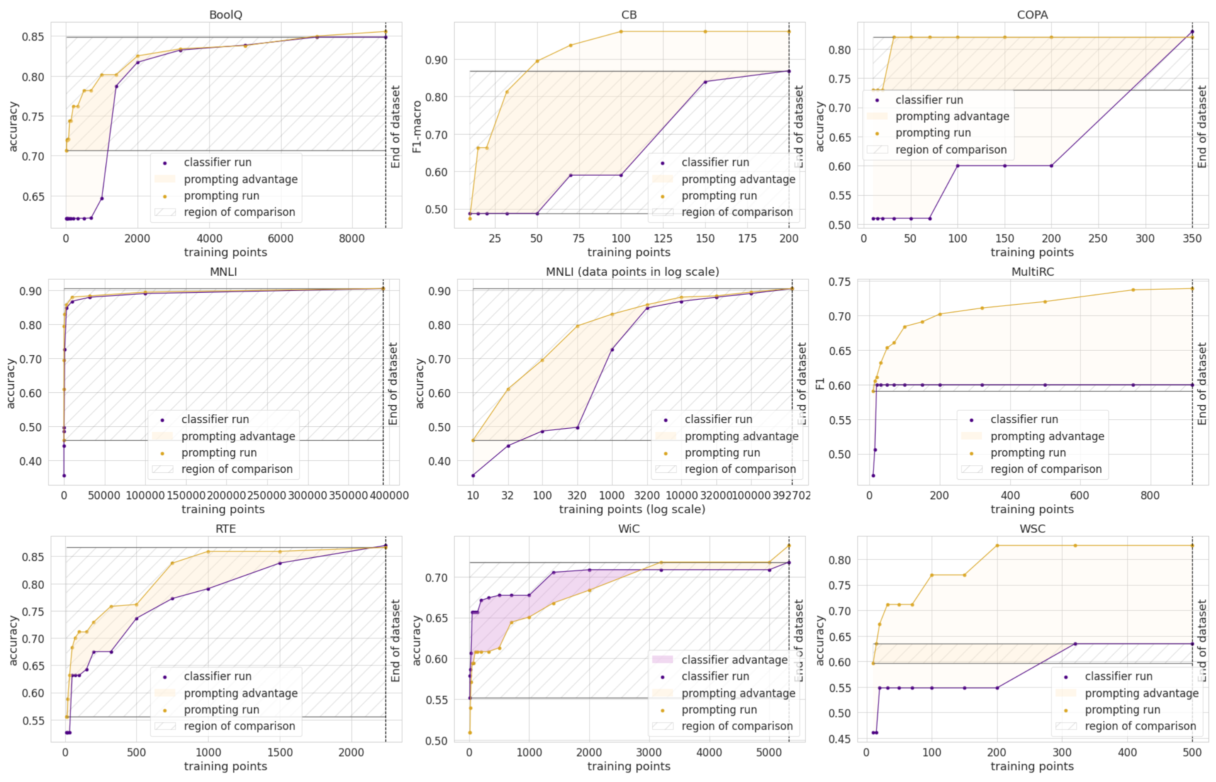


FIGURE 6.1: Prompting vs head (classifier) performance across data scales, up to the full dataset, for seven SuperGLUE tasks & MNLI. Compares the best prompt and head performance at each level of training data across 4 runs. Highlighted region shows the accuracy difference of the models. Cross-hatch region highlights the lowest- and highest- accuracy matched region in the curves. The highlighted area in this region is used to estimate the data advantage from prompting.

WiC as is reported in previous results [Schick and Schütze, 2020b]. Both approaches improve with more training data, but prompting remains better by a varying amount. Many tasks in SuperGLUE have relatively few data points, but we also see an advantage in large datasets like BoolQ and MNLI.

To quantify how many data points the prompt is worth, we first isolate the y-axis band of the lowest- and highest- accuracy where the two curves match in accuracy.[3] The horizontal line at these points represents the advantage of prompting. We then take the integral in this region, i.e. area between the linearly-interpolated curves[4], divided by the height of the band. The area has the dimension of a quantity of data points times the metric unit, so dividing by the performance range yields a # of data points advantage. As low data training is sensitive to noise, in addition to following best training practices we run several different experiments for each x-point. We use a bootstrapping approach to estimate confidence over these runs. Specifically, we hold out one of the 4 head runs and 4 prompt runs (16 combinations total), and compute the standard deviation of those outcomes.

We report these quantities for every task in Table 6.1 as *Average advantage*. For almost all the tasks, we see that prompting gives a substantial advantage in terms of data efficiency, adding the equivalent of hundreds of data points on average.

[3]We assume asymptotically the two curves would match, but are limited by data.
[4]In areas where the head model is better, if any, get subtracted from the total.

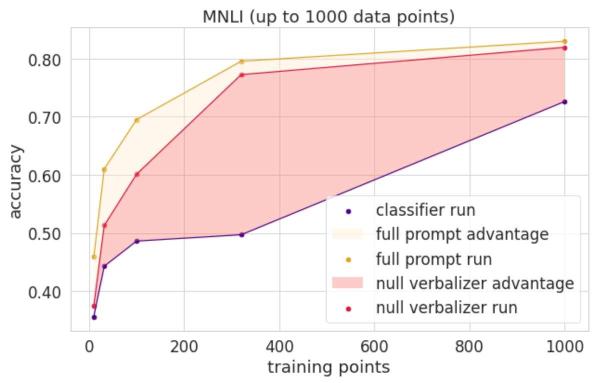
	Average Advantage (# Training Points)							
	MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
P *vs* H	3506 ± 536	752 ± 46	90 ± 2	288 ± 242	384 ± 378	282 ± 34	-424 ± 74	281 ± 137
P *vs* N	150 ± 252	299 ± 81	78 ± 2	-	74 ± 56	404 ± 68	-354 ± 166	-
N *vs* H	3355 ± 612	453 ± 90	12 ± 1	-	309 ± 320	-122 ± 62	-70 ± 160	-

TABLE 6.1: Average prompting advantage in number of data points for MNLI & SuperGLUE tasks. *P* denotes the prompt model, *H* the head model. On average across performance levels, an MNLI prompt model yields the results of an MNLI head model trained with 3500 additional data points. Confidence levels are based on a multiple random runs (see text). *N* indicates a null-verbalizer prompting task that replaces the verbalizer with a non-sensical mapping. *The comparison band of MultiRC is too small as the head baseline fails to learn beyond majority class; we use the full region for a lower-bound result.

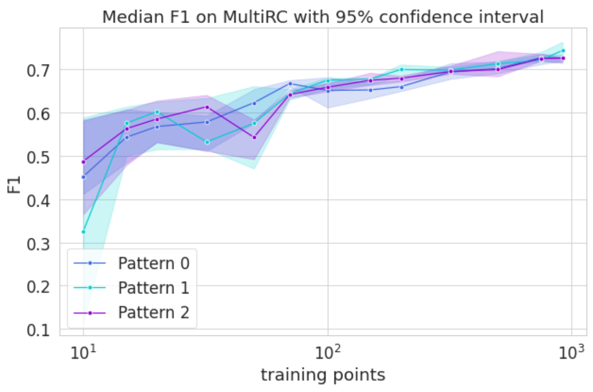
6.6 Analysis

Impact of Pattern vs Verbalizer The intuition of prompts is that they introduce a task description in natural language, even with few training points. To better understand the zero-shot versus adaptive nature of prompts, we consider a *null verbalizer*, a control with a verbalizer that cannot yield semantic information without training. For every task that requires filling in one word (which excludes the more free-form COPA and WSC), we replace the verbalizers, for example, "yes", "no", "maybe", "right" or "wrong", with random first names.

Table 6.1 shows the advantage of the standard prompt over the null verbalizer to estimate this control. We see that for small data tasks such as CB, the null verbalizer removes much of the benefits of prompting. However, with more training data, the model seems to adapt the verbalizer while still gaining the inductive bias benefits of the pattern. Figure 6.2a showcases this dynamic on MNLI. This result further shows that prompting yields data efficiency even if it is not directly analogous to the generation process of training.



(A) Comparison of full prompt and null verbalizer advantage on MNLI at lower data scales.



(B) BLOOM

FIGURE 6.2: Median performance on MultiRC across runs for three prompts. Differences are inconsistent and eclipsed by the variance within one prompt's runs.

Impact of Different Prompts If the prompt acts as a description of the task, one would expect different valid descriptions to vary in their benefits. In order to compare the different prompts we used on each task, we chart the median performance for each of them under different runs. In nearly every experiment, we find that the confidence intervals of those curves largely overlap, implying that prompt choice is not a dominant hyperparameter, i.e. the variance across random seeds usually outweighs the possible benefits of prompt choice. One exception is the low-data regime of BoolQ, where one of the prompts enjoys a significant few-shot advantage over the others. We plot this curve for all MultiRC in Figure 6.2b.

Metric sensitivity We treat each metric linearly in calculating advantage; alternatively, we could re-parameterize the y axis for each task. This choice does not have a consistent effect for or against prompting. For example, emphasizing gains close to convergence increases prompting advantage on CB and MNLI but decreases it on COPA or BoolQ.

6.7 Conclusion

In this chapter, we investigated the use of prompts by pretrained models through a systematic study of the data advantage of prompted finetuning over classifier-based finetuning. Across tasks, prompting consistently yields a varying improvement throughout the training process, which is equivalent to between 100 and 3500 extra finetuning data points. Analysis shows that prompting is mostly robust to pattern choice, and can even learn without an informative verbalizer. This technique allows us to combine the strengths of both finetuning paradigms, which is especially interesting in languages that have access to strong pretrained models, but only limited amounts of downstream task data. With the increased availability of multilingual and lower-resourced monolingual models such as those described in the previous chapters, this category of languages is bound to expand quickly.

Conclusion

Looking backward

This thesis was concerned with making large language model training more available in lower-resource languages. Thanks to their generalization abilities and their ease of use in natural language, LLMs are both the main paradigm of modern NLP research and an increasingly important industrial and consumer product in the wider world. However, mostly because of data constraints, the frontier of their capabilities is only available in English. This motivates our goal of opening up state-of-the-art research and applications beyond just English, as a key part of democratizing this technology.

The first half of this thesis showcased practical work building large language models in a massively multilingual setting as part of an international research collaboration. Both chapters are dedicated to artifacts that were produced in the process. In chapter 3, we present a multilingual text corpus - referred to as ROOTS - built for language modeling, with clear documentation of the process and tools used for future projects. Then, as detailed in chapter 4, we developed a large language model called BLOOM using the ROOTS corpus. This included a detailed account of the training procedure and evaluation of the model, to develop and disseminate LLM training know-how.

Armed with this experience, our work in the second half aimed at assisting practitioners in lower-resource settings. In Chapter 5, we extended the current scaling paradigm, which assumes infinite data, to data-constrained environments, showed that scaling on multiple epochs is still possible with little loss, and derived simple adjustments to make the most of limited data. In 6, we shifted our focus from pretraining to downstream tasks, as finetuning is still an important tool when smaller, efficient models are needed, or when the task is important enough that it will need to be performed several times with strong performance requirements. We showed that prompts added significant information to finetuning setups, which is particularly important when downstream task data is scarce.

Looking forward

What is next for language modeling in low-resource languages? One key question is whether they are better served by inclusion into large massively multilingual models, or by smaller monolingual or linguistic-area-specific models. It seems likely that reusing some of the already existing English-centric resources and models will help, but the best way to do so remains to be seen (Finetuning an existing English LLM? Incorporating large amounts of the target language or languages at pretraining? Training a massively multilingual model, then distilling into an array of target languages?), and the research and practitioner communities in those languages, who tend to be strongly in favor of monolingual models, still need to be convinced that they will be served correctly this way.

Another goal of our work was to democratize LLM training and research. At the time, we hoped that the release of a powerful multilingual language model would unlock new applications and research directions for large language models and that documenting our experience would help the machine learning research community organize new large-scale collaborative projects similar to BigScience. Since then, the experience of BLOOM was useful to create even better language models, such as Mistral-7B [Jiang et al., 2023] or Falcon [Penedo et al., 2023], the open-source finetuning community has become a force in its own right [Chiang et al., 2023, Zheng et al., 2023], and more collaborative communities, such as LAION or AYA, have formed to pool resources and talent and create consensus around large-scale models. In the meantime, proprietary models have also made massive progress [OpenAI, 2023], setting new objectives and fueling a race dynamic between open-source and closed-source models which we see as a key ingredient of the progress of the field.

And what is there to look forward to? Our very speculative ideal is to extend the base of knowledge they can train from as wide as possible, by integrating ever more data in as many modalities as possible, with a special interest for data that brings together several modalities at the same time: videos with subtitles and descriptions, songs and their sheet music and lyrics, times series from physical and social phenomena and the underlying equations... Multilinguality will play a key role in this by multiplying not only the amount of data but also the different points of view available. The hope, then, is that scaling and compute trends continue far out into the future and that sufficiently powerful models will be able to learn all of the latent rules underlying this world of information.

Bibliography

J. Abadji, P. J. Ortiz Suárez, L. Romary, and B. Sagot. Ungoliant: An Optimized Pipeline for the Generation of a Very Large-Scale Multilingual Web Corpus. In *CMLC 2021 - 9th Workshop on Challenges in the Management of Large Corpora*, Limerick / Virtual, Ireland, July 2021. doi: 10.14618/ids-pub-10468. URL https://hal.inria.fr/hal-03301590.

A. Abdelali, F. Guzman, H. Sajjad, and S. Vogel. The amara corpus: Building parallel language resources for the educational domain. In N. C. C. Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA). ISBN 978-2-9517408-8-4.

J. Ács. Exploring bert's vocabulary, 2019. URL http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html.

D. I. Adelani, J. Abbott, G. Neubig, D. D'souza, J. Kreutzer, C. Lignos, C. Palen-Michel, H. Buzaaba, S. Rijhwani, S. Ruder, S. Mayhew, I. A. Azime, S. H. Muhammad, C. C. Emezue, J. Nakatumba-Nabende, P. Ogayo, A. Anuoluwapo, C. Gitau, D. Mbaye, J. Alabi, S. M. Yimam, T. R. Gwadabe, I. Ezeani, R. A. Niyongabo, J. Mukiibi, V. Otiende, I. Orife, D. David, S. Ngom, T. Adewumi, P. Rayson, M. Adeyemi, G. Muriuki, E. Anebi, C. Chukwuneke, N. Odu, E. P. Wairagala, S. Oyerinde, C. Siro, T. S. Bateesa, T. Oloyede, Y. Wambui, V. Akinode, D. Nabagereka, M. Katusiime, A. Awokoya, M. MBOUP, D. Gebreyohannes, H. Tilaye, K. Nwaike, D. Wolde, A. Faye, B. Sibanda, O. Ahia, B. F. P. Dossou, K. Ogueji, T. I. DIOP, A. Diallo, A. Akinfaderin, T. Marengereke, and S. Osei. MasakhaNER: Named entity recognition for African languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131, 2021. doi: 10.1162/tacl_a_00416. URL https://aclanthology.org/2021.tacl-1.66.

A. Aghajanyan, D. Okhonko, M. Lewis, M. Joshi, H. Xu, G. Ghosh, and L. Zettlemoyer. HTLM: Hypertext pre-training and prompting of language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=P-pPW1nxf1r.

A. Aghajanyan, L. Yu, A. Conneau, W.-N. Hsu, K. Hambardzumyan, S. Zhang, S. Roller, N. Goyal, O. Levy, and L. Zettlemoyer. Scaling laws for generative mixed-modal language models. *arXiv preprint arXiv:2301.03728*, 2023.

Ž. Agić and I. Vulić. JW300: A wide-coverage parallel corpus for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3204–3210, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1310. URL https://aclanthology.org/P19-1310.

I. M. Alabdulmohsin, B. Neyshabur, and X. Zhai. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.

J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

M. Allamanis. The adverse effects of code duplication in machine learning models of code. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 143–153, 2019.

M. Allamanis, E. T. Barr, C. Bird, and C. Sutton. Suggesting accurate method and class names. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 38–49, 2015.

M. Alrabiah, A. Alsalman, and E. Atwell. The design and construction of the 50 million words ksucca king saud university corpus of classical arabic. 01 2013.

M. Aly and A. Atiya. LABR: A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics. URL https://aclanthology.org/P13-2088.

Z. Alyafeai, M. Masoud, M. Ghaleb, and M. S. AlShaibani. Masader: Metadata sourcing for arabic text and speech data resources. *CoRR*, abs/2110.06744, 2021. URL https://arxiv.org/abs/2110.06744.

J. Armengol-Estapé, C. P. Carrino, C. Rodriguez-Penagos, O. de Gibert Bonet, C. Armentano-Oller, A. Gonzalez-Agirre, M. Melero, and M. Villegas. Are multilingual models the best choice for moderately under-resourced languages? A comprehensive assessment for Catalan. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4933–4946, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.437. URL https://aclanthology.org/2021.findings-acl.437.

M. Artetxe, I. Aldabe, R. Agerri, O. Perez-de Viñaspre, and A. Soroa. Does corpus quality really matter for low-resource languages?, 2022. URL https://arxiv.org/abs/2203.08111.

A. Ashari. Indonesian news articles published at 2017, 2018. URL https://www.kaggle.com/datasets/aashari/indonesian-news-articles-published-at-2017.

S. Bach, V. Sanh, Z. X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Fevry, Z. Alyafeai, M. Dey, A. Santilli, Z. Sun, S. Ben-david, C. Xu, G. Chhablani, H. Wang, J. Fries, M. Alshaibani, S. Sharma, U. Thakker, K. Almubarak, X. Tang, D. Radev, M. T.-j. Jiang, and A. Rush. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.9. URL https://aclanthology.org/2022.acl-demo.9.

S. H. Bach, V. Sanh, Z.-X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Fevry, Z. Alyafeai, M. Dey, A. Santilli, Z. Sun, S. Ben-David, C. Xu, G. Chhablani, H. Wang, J. A. Fries, M. S. Al-shaibani, S. Sharma, U. Thakker, K. Almubarak, X. Tang, X. Tang, M. T.-J. Jiang, and A. M. Rush. Promptsource: An integrated development environment and repository for natural language prompts, 2022b.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016.

L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, 1983. doi: 10.1109/TPAMI.1983.4767370.

Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

J. Bandy and N. Vincent. Addressing" documentation debt" in machine learning research: A retrospective datasheet for bookcorpus. *arXiv preprint arXiv:2105.05241*, 2021.

Y. Bansal, B. Ghorbani, A. Garg, B. Zhang, C. Cherry, B. Neyshabur, and O. Firat. Data scaling laws in nmt: The effect of noise and architecture. In *International Conference on Machine Learning*, pages 1466–1482. PMLR, 2022.

R. Bawden and F. Yvon. Investigating the translation performance of a large multilingual language model: the case of BLOOM. *CoRR*, abs/2303.01911, 2023. doi: 10.48550/arXiv.2303.01911. URL https://doi.org/10.48550/arXiv.2303.01911.

R. Bawden, E. Bilinski, T. Lavergne, and S. Rosset. DiaBLa: A Corpus of Bilingual Spontaneous Written Dialogues for Machine Translation. *Language Resources and Evaluation*, pages 635–660, 2020. doi: 10.1007/s10579-020-09514-4. URL https://doi.org/10.1007/s10579-020-09514-4.

Y. Belinkov, A. Magidow, A. Barrón-Cedeño, A. Shmidman, and M. Romanov. Studying the history of the arabic language: language technology and a large-scale historical corpus. *Language Resources and Evaluation*, 53(4):771–805, 2019.

E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

Z. Bian, H. Liu, B. Wang, H. Huang, Y. Li, C. Wang, F. Cui, and Y. You. Colossal-ai: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*, 2021.

S. Biderman, K. Bicheno, and L. Gao. Datasheet for the pile. *arXiv preprint arXiv:2201.07311*, 2022.

S. Biderman, U. S. Prashanth, L. Sutawika, H. Schoelkopf, Q. Anthony, S. Purohit, and E. Raf. Emergent and predictable memorization in large language models. *arXiv preprint arXiv:2304.11158*, 2023.

BigScience Workshop. BLOOM (revision 4ab0472), 2022. URL https://huggingface.co/bigscience/bloom.

S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58, 2021.

S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode\# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, 2022.

S. L. Blodgett, G. Lopez, A. Olteanu, R. Sim, and H. Wallach. Stereotyping Norwegian salmon: An inventory of pitfalls in fairness benchmark datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1004–1015, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.81. URL https://aclanthology.org/2021.acl-long.81.

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302. URL https://aclanthology.org/W14-3302.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992. URL https://aclanthology.org/J92-4003.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Budiono, H. Riza, and C. Hakim. Resource report: Building parallel text corpora for multi-domain translation system. In *Proceedings of the 7th Workshop on Asian Language Resources (ALR7)*, pages 92–95, Suntec, Singapore, Aug. 2009. Association for Computational Linguistics. URL https://aclanthology.org/W09-3413.

V. Q. Bình. Binhvq news corpus. https://github.com/binhvq/news-corpus, 2021.

N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

C. P. Carrino, C. G. Rodriguez-Penagos, and C. Armentano-Oller. Tecla: Text classification catalan dataset, Mar. 2021. URL https://doi.org/10.5281/zenodo.4761505.

T. Castro Ferreira, C. Gardent, N. Ilinykh, C. van der Lee, S. Mille, D. Moussallem, and A. Shimorina. The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, pages 55–76, Dublin, Ireland (Virtual), 2020. Association for Computational Linguistics.

J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, and J. Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

M. Cettolo, C. Girardi, and M. Federico. WIT3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Annual conference of the European Association for Machine Translation*, pages 261–268, Trento, Italy, May 28–30 2012. European Association for Machine Translation. URL https://www.aclweb.org/anthology/2012.eamt-1.60.

B. Chan, S. Schweter, and T. Möller. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.598. URL https://aclanthology.org/2020.coling-main.598.

M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 380–388, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134959. doi: 10.1145/509907.509965. URL https://doi.org/10.1145/509907.509965.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Y. Chen and A. Eisele. MultiUN v2: UN documents with multilingual alignments. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2500–2504, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/641_Paper.pdf.

W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022. URL https://arxiv.org/abs/2210.11416.

C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

C. B. Clement, M. Bierbaum, K. P. O'Keeffe, and A. A. Alemi. On the use of arxiv as a dataset. *CoRR*, abs/1905.00075, 2019. URL http://arxiv.org/abs/1905.00075.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

D. Contractor, D. McDuff, J. K. Haines, J. Lee, C. Hines, B. Hecht, N. Vincent, and H. Li. Behavioral use licensing for responsible ai. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 778–788, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393522. doi: 10.1145/3531146.3533143. URL https://doi.org/10.1145/3531146.3533143.

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

A. M. Dai and Q. V. Le. Semi-supervised sequence learning. *CoRR*, abs/1511.01432, 2015. URL http://arxiv.org/abs/1511.01432.

D. David. Swahili: News classification dataset, Dec. 2020. URL https://doi.org/10.5281/zenodo.4300294.

M.-C. De Marneffe, M. Simons, and J. Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124, 2019.

T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020.

J. Dodge, M. Sap, A. Marasović, W. Agnew, G. Ilharco, D. Groeneveld, M. Mitchell, and M. Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013. URL http://arxiv.org/abs/1310.1531.

D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

O. Einea, A. Elnagar, and R. Al Debsi. Sanad: Single-label arabic news articles dataset for automatic text categorization. *Data in Brief*, 25:104076, 2019. ISSN 2352-3409. doi: https://doi.org/10.1016/j.dib.2019.104076. URL https://www.sciencedirect.com/science/article/pii/S2352340919304305.

M. El-Haj. Habibi - a multi dialect multi national Arabic song lyrics corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1318–1326, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.165.

M. El-Haj and R. Koulali. Kalimat a multipurpose arabic corpus. In *Second workshop on Arabic corpus linguistics (WACL-2)*, pages 22–25, 2013.

I. A. El-Khair. 1.5 billion words arabic corpus. *arXiv preprint arXiv:1611.04033*, 2016.

A. Elnagar, L. Lulu, and O. Einea. An annotated huge dataset for standard and colloquial arabic reviews for subjective sentiment analysis. *Procedia Computer Science*, 142:182–189, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.10.474. URL https://www.sciencedirect.com/science/article/pii/S1877050918321781. Arabic Computational Linguistics.

A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary, N. Goyal, T. Birch, V. Liptchinsky, S. Edunov, M. Auli, and A. Joulin. Beyond English-Centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48, 2021. URL http://jmlr.org/papers/v22/20-1307.html.

Y. Fathullah, C. Wu, E. Lakomkin, J. Jia, Y. Shangguan, K. Li, J. Guo, W. Xiong, J. Mahadeokar, O. Kalinli, C. Fuegen, and M. Seltzer. Prompting large language models with speech recognition abilities, 2023.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

J. FitzGerald, C. Hench, C. Peris, S. Mackie, K. Rottmann, A. Sanchez, A. Nash, L. Urbach, V. Kakarala, R. Singh, S. Ranganath, L. Crist, M. Britan, W. Leeuwis, G. Tur, and P. Natarajan. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages, 2022. URL https://arxiv.org/abs/2204.08582.

D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih, L. Zettlemoyer, and M. Lewis. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*, 2022.

D. Y. Fu, T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=COZDy0WYGg.

P. Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, feb 1994. ISSN 0898-9788.

S. Galliano, E. Geoffrois, G. Gravier, J.-F. Bonastre, D. Mostefa, and K. Choukri. Corpus description of the ESTER evaluation campaign for the rich transcription of French broadcast news. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2006/pdf/646_pdf.pdf.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020a. URL https://arxiv.org/abs/2101.00027.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020b.

L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, Sept. 2021. URL https://doi.org/10.5281/zenodo.5371628.

S. Gehrmann, T. Adewumi, K. Aggarwal, P. S. Ammanamanchi, A. Anuoluwapo, A. Bosselut, K. R. Chandu, M. Clinciu, D. Das, K. D. Dhole, et al. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*, 2021.

S. Gehrmann, A. Bhattacharjee, A. Mahendiran, A. Wang, A. Papangelis, A. Madaan, A. McMillan-Major, A. Shvets, A. Upadhyay, B. Yao, B. Wilie, C. Bhagavatula, C. You, C. Thomson, C. Garbacea, D. Wang, D. Deutsch, D. Xiong, D. Jin, D. Gkatzia, D. Radev, E. Clark, E. Durmus, F. Ladhak, F. Ginter, G. I. Winata, H. Strobelt, H. Hayashi, J. Novikova, J. Kanerva, J. Chim, J. Zhou, J. Clive, J. Maynez, J. Sedoc, J. Juraska,

K. Dhole, K. R. Chandu, L. Perez-Beltrachini, L. F. R. Ribeiro, L. Tunstall, L. Zhang, M. Pushkarna, M. Creutz, M. White, M. S. Kale, M. K. Eddine, N. Daheim, N. Subramani, O. Dusek, P. P. Liang, P. S. Ammanamanchi, Q. Zhu, R. Puduppully, R. Kriz, R. Shahriyar, R. Cardenas, S. Mahamood, S. Osei, S. Cahyawijaya, S. Štajner, S. Montella, Shailza, S. Jolly, S. Mille, T. Hasan, T. Shen, T. Adewumi, V. Raunak, V. Raheja, V. Nikolaev, V. Tsai, Y. Jernite, Y. Xu, Y. Sang, Y. Liu, and Y. Hou. Gemv2: Multilingual nlg benchmarking in a single line of code, 2022a. URL https://arxiv.org/abs/2206.11249.

S. Gehrmann, E. Clark, and T. Sellam. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text, 2022b. URL https://arxiv.org/abs/2202.06935.

B. Ghorbani, O. Firat, M. Freitag, A. Bapna, M. Krikun, X. Garcia, C. Chelba, and C. Cherry. Scaling laws for neural machine translation. *arXiv preprint arXiv:2109.07740*, 2021.

R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

A. Gokaslan and V. Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

D. Goldhahn, T. Eckart, and U. Quasthoff. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/327_Paper.pdf.

Y. Gong, H. Luo, A. H. Liu, L. Karlinsky, and J. Glass. Listen, think, and understand, 2023.

N. Goyal, C. Gao, V. Chaudhary, P.-J. Chen, G. Wenzek, D. Ju, S. Krishnan, M. Ranzato, F. Guzmán, and A. Fan. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022. doi: 10.1162/tacl_a_00474. URL https://aclanthology.org/2022.tacl-1.30.

A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33:1474–1487, 2020.

A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

S. Gururaja, A. Bertsch, C. Na, D. G. Widder, and E. Strubell. To build our future, we must know our past: Contextualizing paradigm shifts in natural language processing, 2023.

P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654, 2020. URL https://arxiv.org/abs/2006.03654.

W. He, K. Liu, J. Liu, Y. Lyu, S. Zhao, X. Xiao, Y. Liu, Y. Wang, H. Wu, Q. She, X. Liu, T. Wu, and H. Wang. Dureader: a chinese machine reading comprehension dataset from real-world applications, 2018.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding, 2021.

T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

D. Hernandez, T. Brown, T. Conerly, N. DasSarma, D. Drain, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, T. Henighan, T. Hume, S. Johnston, B. Mann, C. Olah, C. Olsson, D. Amodei, N. Joseph, J. Kaplan, and S. McCandlish. Scaling laws and interpretability of learning from repeated data, 2022. URL https://arxiv.org/abs/2201.10066.

J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

V. A. Ho, D. H.-C. Nguyen, D. H. Nguyen, L. T.-V. Pham, D.-V. Nguyen, K. V. Nguyen, and N. L.-T. Nguyen. Emotion recognition for vietnamese social media text, 2020.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022a. URL https://arxiv.org/abs/2203.15556.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022b.

J. Howard and S. Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018. URL http://arxiv.org/abs/1801.06146.

S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, 2016.

B. Jawaid, A. Kamran, and O. Bojar. Urdu monolingual corpus. 2014.

C. Jia, Y. Yang, Y. Xia, Y. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *CoRR*, abs/2102.05918, 2021. URL https://arxiv.org/abs/2102.05918.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. Le Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

Z. Jiang, F. F. Xu, J. Araki, and G. Neubig. How can we know what language models know?, 2020.

D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, and P. Dubey. A study of bfloat16 for deep learning training, 2019.

N. Kandpal, E. Wallace, and C. Raffel. Deduplicating training data mitigates privacy risks in language models, 2022.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

H. Kermes, S. Degaetano-Ortlieb, A. Khamis, J. Knappen, and E. Teich. The royal society corpus: From uncharted data to corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1928–1931, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1305.

D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In M. A. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 252–262. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1023. URL https://doi.org/10.18653/v1/n18-1023.

M. Khrushchev, R. Vasilev, A. Petrov, and N. Zinov. YaLM 100B, 6 2022. URL https://github.com/yandex/YaLM-100B.

B. Kim, H. Kim, S.-W. Lee, G. Lee, D. Kwak, J. Dong Hyeon, S. Park, S. Kim, S. Kim, D. Seo, H. Lee, M. Jeong, S. Lee, M. Kim, S. H. Ko, S. Kim, T. Park, J. Kim, S. Kang, N.-H. Ryu, K. M. Yoo, M. Chang, S. Suh, S. In, J. Park, K. Kim, H. Kim, J. Jeong, Y. G. Yeo, D. Ham, D. Park, M. Y. Lee, J. Kang, I. Kang, J.-W. Ha, W. Park, and N. Sung. What changes can large-scale language models bring? intensive study on HyperCLOVA: Billions-scale korean generative pretrained transformers. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

W. Klöpffer. Life cycle assessment. *Environmental Science and Pollution Research*, 4(4):223–228, 1997.

D. Kocetkov, R. Li, L. B. Allal, J. Li, C. Mou, C. M. Ferrandis, Y. Jernite, M. Mitchell, S. Hughes, T. Wolf, et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

A. Komatsuzaki. One epoch is all you need. *arXiv preprint arXiv:1906.06669*, 2019.

M. Kowsher, M. Uddin, A. Tahabilder, M. Ruhul Amin, M. F. Shahriar, and M. S. I. Sobuj. Banglalm: Bangla corpus for language model research. Online, September 2021. IEEE. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3882903.

J. Kreutzer, I. Caswell, L. Wang, A. Wahab, D. van Esch, N. Ulzii-Orshikh, A. Tapo, N. Subramani, A. Sokolov, C. Sikasote, M. Setyawan, S. Sarin, S. Samb, B. Sagot, C. Rivera, A. Rios, I. Papadimitriou, S. Osei, P. O. Suarez, I. Orife, K. Ogueji, R. Niyongabo, T. Nguyen, M. Müller, A. Müller, S. Muhammad, N. Muhammad, A. Mnyakeni, J. Mirzakhalov, T. Matangira, C. Leong, N. Lawson, S. Kudugunta,

Y. Jernite, M. Jenny, O. Firat, B. Dossou, S. Dlamini, N. de Silva, S. Çabuk Ballı, S. Biderman, A. Battisti, A. Baruwa, A. Bapna, P. Baljekar, I. Azime, A. Awokoya, D. Ataman, O. Ahia, O. Ahia, S. Agrawal, and M. Adeyemi. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10(0):50–72, 2022. ISSN 2307-387X. URL https://transacl.org/index.php/tacl/article/view/3317.

T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

A. Kunchukuttan, P. Mehta, and P. Bhattacharyya. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1548.

A. Kunchukuttan, D. Kakwani, S. Golla, C. GokulN., A. Bhattacharyya, M. M. Khapra, and P. Kumar. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages. *ArXiv*, abs/2005.00085, 2020.

K. Kurniawan and S. Louvan. Indosum: A new benchmark dataset for indonesian text summarization. In *2018 International Conference on Asian Language Processing (IALP)*, pages 215–220. IEEE, 2018.

B. Külebi. ParlamentParla - Speech corpus of Catalan Parliamentary sessions, Oct. 2021. URL https://doi.org/10.5281/zenodo.5541827.

A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

F. Ladhak, E. Durmus, C. Cardie, and K. McKeown. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.360. URL https://aclanthology.org/2020.findings-emnlp.360.

G. Lample and A. Conneau. Cross-lingual language model pretraining, 2019. URL https://arxiv.org/abs/1901.07291.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019. URL http://arxiv.org/abs/1909.11942.

H. Laurençon, L. Saulnier, T. Wang, C. Akiki, A. Villanova del Moral, T. Le Scao, L. Von Werra, C. Mou, E. González Ponferrada, H. Nguyen, et al. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/ce9e92e3de2372a4b93353eb7f3dc0bd-Abstract-Datasets_and_Benchmarks.html.

H. Laurençon, L. Saulnier, L. Tronchon, S. Bekman, A. Singh, A. Lozhkov, T. Wang, S. Karamcheti, A. M. Rush, D. Kiela, M. Cord, and V. Sanh. Obelics: An open web-scale filtered dataset of interleaved image-text documents, 2023.

H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, and D. Schwab. Flaubert: Unsupervised language model pre-training for french. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France, May 2020. European Language Resources Association. URL https://www.aclweb.org/anthology/2020.lrec-1.302.

T. Le Scao and A. Rush. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021. URL https://aclanthology.org/2021.naacl-main.208.

T. Le Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022a. URL https://arxiv.org/abs/2211.05100.

T. Le Scao, T. Wang, D. Hesslow, L. Saulnier, S. Bekman, M. Saiful Bari, S. Biderman, H. Elsahar, J. Phang, O. Press, et al. What language model to train if you have one million GPU hours? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022b. URL https://aclanthology.org/2022.findings-emnlp.54/.

C. Lee, K. Cho, and W. Kang. Mixout: Effective regularization to finetune large-scale pretrained language models, 2020.

K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2022.

B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *CoRR*, abs/2104.08691, 2021. URL https://arxiv.org/abs/2104.08691.

H. J. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, 2012. URL http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4492.

A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra. Solving quantitative reasoning problems with language models, 2022.

Q. Lhoest, A. Villanova del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. Le Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. Rush, and T. Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican

Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-demo.21. URL https://aclanthology.org/2021.emnlp-demo.21.

R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023a.

X. Li, T. Zhang, Y. Dubois, R. Taori, I. Gulrajani, C. Guestrin, P. Liang, and T. B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023b.

X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. d. M. d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals. Competition-level code generation with alphacode, 2022. URL https://arxiv.org/abs/2203.07814.

P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. Orr, L. Zheng, M. Yuksekgonul, M. Suzgun, N. Kim, N. Guha, N. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, and Y. Koreeda. Holistic evaluation of language models, 2022a. URL https://arxiv.org/abs/2211.09110.

P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022b.

O. Lieber, O. Sharir, B. Lenz, and Y. Shoham. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs*, 2021.

C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

X. V. Lin, T. Mihaylov, M. Artetxe, T. Wang, S. Chen, D. Simig, M. Ott, N. Goyal, S. Bhosale, J. Du, R. Pasunuru, S. Shleifer, P. S. Koura, V. Chaudhary, B. O'Horo, J. Wang, L. Zettlemoyer, Z. Kozareva, M. Diab, V. Stoyanov, and X. Li. Few-shot learning with multilingual language models, 2021a. URL https://arxiv.org/abs/2112.10668.

X. V. Lin, T. Mihaylov, M. Artetxe, T. Wang, S. Chen, D. Simig, M. Ott, N. Goyal, S. Bhosale, J. Du, et al. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*, 2021b.

P. Lison and J. Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. 2016.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.447. URL https://www.aclweb.org/anthology/2020.acl-main.447.

C. V. Lopes, P. Maj, P. Martins, V. Saini, D. Yang, J. Zitny, H. Sajnani, and J. Vitek. Déjàvu: a map of code duplicates on github. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–28, 2017.

I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL http://arxiv.org/abs/1608.03983.

A. S. Luccioni and J. D. Viviano. What's in the box? a preliminary analysis of undesirable content in the common crawl corpus. *Published in the Proceedings of ACL 2021*, 2021.

A. S. Luccioni, S. Viguier, and A.-L. Ligozat. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *arXiv preprint arXiv:2211.02001*, 2022.

H. T. Madabushi, E. Gow-Smith, M. Garcia, C. Scarton, M. Idiart, and A. Villavicencio. Semeval-2022 task 2: Multilingual idiomaticity detection and sentence embedding. *arXiv preprint arXiv:2204.10050*, 2022.

R. Mahendra, A. F. Aji, S. Louvan, F. Rahman, and C. Vania. Indonli: A natural language inference dataset for indonesian. *arXiv preprint arXiv:2110.14566*, 2021.

U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993. doi: 10.1137/0222058. URL https://doi.org/10.1137/0222058.

G. S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 141–150, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242592. URL https://doi.org/10.1145/1242572.1242592.

L. Martin, B. Muller, P. J. Ortiz Suárez, Y. Dupont, L. Romary, É. de la Clergerie, D. Seddah, and B. Sagot. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the*

Association for Computational Linguistics, pages 7203–7219, Online, July 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.acl-main.645.

T. T. Mayeesha, A. M. Sarwar, and R. M. Rahman. Deep learning based question answering system in Bengali, Nov. 2020. URL https://doi.org/10.5281/zenodo.4557874.

B. McCann, N. S. Keskar, C. Xiong, and R. Socher. The natural language decathlon: Multitask learning as question answering, 2018.

R. T. McCoy, J. Min, and T. Linzen. Berts of a feather do not generalize together: Large variability in generalization across models with similar test set performance, 2019.

A. McMillan-Major, Z. Alyafeai, S. Biderman, K. Chen, F. De Toni, G. Dupont, H. Elsahar, C. Emezue, A. F. Aji, S. Ilić, N. Khamis, C. Leong, M. Masoud, A. Soroa, P. O. Suarez, Z. Talat, D. van Strien, and Y. Jernite. Documenting geographically and contextually diverse data sources: The bigscience catalogue of language data and resources, 2022. URL https://arxiv.org/abs/2201.10066.

MFAIRD. Human-level play in the game of <i>diplomacy</i> by combining language models with strategic reasoning. *Science*, 2022. doi: 10.1126/science.ade9097.

P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

S. J. Mielke, Z. Alyafeai, E. Salesky, C. Raffel, M. Dey, M. Gallé, A. Raja, C. Si, W. Y. Lee, B. Sagot, and S. Tan. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp, 2021. URL https://arxiv.org/abs/2112.10508.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 220–229, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361255. doi: 10.1145/3287560.3287596. URL https://doi.org/10.1145/3287560.3287596.

D. Moeljadi. Usage of indonesian possessive verbal predicates: A statistical analysis based on questionnaire and storytelling surveys. In *5th Conference on Austronesian and Papuan Languages and Linguistics (APLL5)*, SOAS, University of London, 2012.

G. Mohr, J. Kunze, and M. Stack. The warc file format 1.0 (iso 28500). 2008.

A. Moi, P. Cistac, N. Patry, E. P. Walsh, F. Morgan, S. Pütz, T. Wolf, S. Gugger, C. Delangue, J. Chaumond, L. Debut, and P. von Platen. Hugging face tokenizers library. https://github.com/huggingface/tokenizers, 2019.

R. Moore, J. Dowding, J. Gawron, and D. Moran. Combining linguistic and statistical knowledge sources in natural-language processing for atis. 03 1996.

M. Mosbach, M. Andriushchenko, and D. Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines, 2020.

N. Muennighoff. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*, 2022.

N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022a. doi: 10.48550/ARXIV.2210.07316. URL https://arxiv.org/abs/2210.07316.

N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, et al. Crosslingual generalization through multitask finetuning. 2022b.

N. Muennighoff, A. M. Rush, B. Barak, T. Le Scao, A. Piktus, N. Tazi, S. Pyysalo, T. Wolf, and C. Raffel. Scaling data-constrained language models. In *the coming Thirty-sixth Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=j5BuTrEj35.

S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021 (12):124003, 2021.

N. Nangia, C. Vania, R. Bhalerao, and S. R. Bowman. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.154. URL https://aclanthology.org/2020.emnlp-main.154.

S. Narang, H. W. Chung, Y. Tay, W. Fedus, T. Fevry, M. Matena, K. Malkan, N. Fiedel, N. Shazeer, Z. Lan, Y. Zhou, W. Li, N. Ding, J. Marcus, A. Roberts, and C. Raffel. Do transformer modifications transfer across implementations and applications? In *Conference on Empirical Methods in Natural Language Processing*, 2021.

D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

G. Neubig. How large is a 'large' language model?, 2023. URL https://twitter.com/gneubig/status/1631386071228358658.

A. Névéol, Y. Dupont, J. Bezançon, and K. Fort. French CrowS-pairs: Extending a challenge dataset for measuring social bias in masked language models to a language other than English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8521–8531, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.583. URL https://aclanthology.org/2022.acl-long.583.

C. Ngo and T. H. Trinh. Styled augmented translation (sat). *https://github.com/vietai/SAT*, 2021.

K. V. Nguyen, V. D. Nguyen, P. X. V. Nguyen, T. T. H. Truong, and N. L.-T. Nguyen. Uit-vsfc: Vietnamese students' feedback corpus for sentiment analysis. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, pages 19–24, 2018. doi: 10.1109/KSE.2018.8573337.

T. Nguyen, H. Pham, M. Truong, H. Duc, and P. Tan. Vietnamese poem generator. https://github.com/fsoft-ailab/Poem-Generator, 2021.

E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

J. Nivre, D. Zeman, F. Ginter, and F. Tyers. Universal Dependencies. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-5001.

H. Nomoto, K. Okano, D. Moeljadi, and H. Sawada. Tufs asian language parallel corpus (talpco). In *Proceedings of the Twenty-Fourth Annual Meeting of the Association for Natural Language Processing*, pages 436–439. Association for Natural Language Processing, 2018.

nostalgebraist. chinchilla's wild implications. *lesswrong*, 2022.

OpenAI. Gpt-4 technical report, 2023.

G. Orlanski, K. Xiao, X. Garcia, J. Hui, J. Howland, J. Malmaud, J. Austin, R. Singh, and M. Catasta. Measuring the impact of programming language distribution. *arXiv preprint arXiv:2302.01973*, 2023.

P. J. Ortiz Suárez, B. Sagot, and L. Romary. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9 – 16, Mannheim, 2019. Leibniz-Institut für Deutsche Sprache. doi: 10.14618/ids-pub-9021. URL http://nbn-resolving.de/urn:nbn:de:bsz:mh39-90215.

P. J. Ortiz Suárez, L. Romary, and B. Sagot. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online, July 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.acl-main.156.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023.

M. T. Pilehvar and J. Camacho-Collados. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *CoRR*, abs/1808.09121, 2018. URL http://arxiv.org/abs/1808.09121.

F. Pisceldo, R. Manurung, and M. Adriani. Probabilistic part-of-speech tagging for bahasa indonesia. In *Third International Workshop on Malay and Indonesian Language Engineering (MALINDO)*, Suntec, Singapore, 2009.

M. Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. URL https://aclanthology.org/W18-6319.

O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

R. Puri and B. Catanzaro. Zero-shot text classification with generative language models. *CoRR*, abs/1912.10165, 2019. URL http://arxiv.org/abs/1912.10165.

A. Radford. Improving language understanding by generative pre-training. 2018.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021a.

J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, H. F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. M. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz,

T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d'Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. A. Hechtman, L. Weidinger, I. Gabriel, W. S. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021b. URL https://arxiv.org/abs/2112.11446.

R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

M. Rahman, E. Kumar Dey, et al. Datasets for aspect-based sentiment analysis in bangla and its baseline evaluation. *Data*, 3(2):15, 2018.

F. Rahutomo and A. Miqdad Muadz Muzad. Indonesian news corpus, 2018. URL https://data.mendeley.com/datasets/2zpbjs22k3/1.

S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. ZeRO: Memory optimizations toward training trillion parameter models. *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2020. doi: 10.1109/sc41405.2020.00024. URL http://dx.doi.org/10.1109/SC41405.2020.00024.

D. Raji, E. Denton, E. M. Bender, A. Hanna, and A. Paullada. Ai and the everything in the whole wide world benchmark. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/084b6fbb10729ed4da8c3d3f5a3ae7c9-Paper-round2.pdf.

A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021a. URL https://arxiv.org/abs/2102.12092.

G. Ramesh, S. Doddapaneni, A. Bheemaraj, M. Jobanputra, R. AK, A. Sharma, S. Sahoo, H. Diddee, M. J, D. Kakwani, N. Kumar, A. Pradeep, S. Nagaraj, K. Deepak, V. Raghavan, A. Kunchukuttan, P. Kumar, and M. S. Khapra. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages, 2021b.

J. Rasley, S. Rajbhandari, O. Ruwase, and Y. 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 & Data Mining*, pages 3505–3506, 2020.

S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent, 2022.

C. G. Rodriguez-Penagos and C. Armentano-Oller. Enriched conllu ancora for ml training, June 2021a. URL https://doi.org/10.5281/zenodo.5036651.

C. G. Rodriguez-Penagos and C. Armentano-Oller. VilaQuAD: an extractive QA dataset from Catalan newswire, Feb. 2021b. URL https://doi.org/10.5281/zenodo.4761430.

C. G. Rodriguez-Penagos and C. Armentano-Oller. ViquiQuAD: an extractive QA dataset from Catalan Wikipedia, Feb. 2021c. URL https://doi.org/10.5281/zenodo.4761412.

M. Roemmele, C. A. Bejan, and A. S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI, 2011. URL http://www.aaai.org/ocs/index.php/SSS/SSS11/paper/view/2418.

A. Rogers. Changing the world by changing the data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2182–2194, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.170. URL https://aclanthology.org/2021.acl-long.170.

B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve. Code llama: Open foundation models for code, 2023.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.243. URL https://aclanthology.org/2021.acl-long.243.

A. Safaya, M. Abdullatif, and D. Yuret. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online), Dec. 2020. International Committee for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.semeval-1.271.

H. Sajjad, A. Abdelali, N. Durrani, and F. Dalvi. AraBench: Benchmarking dialectal Arabic-English machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5094–5107, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.447. URL https://aclanthology.org/2020.coling-main.447.

S. Sakti, E. Kelana, H. Riza, S. Sakai, K. Markov, and S. Nakamura. Development of Indonesian large vocabulary continuous speech recognition system within a-STAR project. In *Proceedings of the*

Workshop on Technologies and Corpora for Asia-Pacific Speech Translation (TCAST), 2008. URL https://aclanthology.org/I08-8004.

V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. Le Scao, A. Raja, et al. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*, 2022.

T. Schick and H. Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2020a.

T. Schick and H. Schütze. It's not just size that matters: Small language models are also few-shot learners, 2020b.

T. Schick, H. Schmid, and H. Schütze. Automatically identifying words that can serve as labels for few-shot text classification, 2020.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

M. Schuster and K. Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *Communications of the ACM*, 63(12), 2020.

C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3), 1948.

N. Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

S. C. Shikali and M. Refuoe. Language modeling data for swahili, Nov. 2019. URL https://doi.org/10.5281/zenodo.3553423.

S. Shin, S.-W. Lee, H. Ahn, S. Kim, H. Kim, B. Kim, K. Cho, G. Lee, W. Park, J.-W. Ha, et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*, 2022.

T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *CoRR*, abs/2010.15980, 2020. URL https://arxiv.org/abs/2010.15980.

O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova, and T. Shavrina. mgpt: Few-shot learners go multilingual. *arXiv preprint arXiv:2204.07580*, 2022.

M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

A. Simoulin and B. Crabbé. Un modèle Transformer Génératif Pré-entrainé pour le ______ français. In P. Denis, N. Grabar, A. Fraisse, R. Cardon, B. Jacquemin, E. Kergosien, and A. Balvet, editors, *Traitement Automatique des Langues Naturelles*, pages 246–255, Lille, France, 2021. ATALA. URL https://hal.archives-ouvertes.fr/hal-03265900.

S. Siripragada, J. Philip, V. P. Namboodiri, and C. V. Jawahar. A multilingual parallel corpora collection effort for Indian languages. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3743–3751, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.462.

S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls, A. Rosenbaum, A. Rumshisky, C. S. Prakash, M. Sridhar, F. Triefenbach, A. Verma, G. Tur, and P. Natarajan. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model, 2022a. URL https://arxiv.org/abs/2208.01448.

S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls, A. Rosenbaum, A. Rumshisky, et al. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *arXiv preprint arXiv:2208.01448*, 2022b.

B. Sorscher, R. Geirhos, S. Shekhar, S. Ganguli, and A. S. Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*, 2022.

A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

H. Su, X. Zhou, H. Yu, Y. Chen, Z. Zhu, Y. Yu, and J. Zhou. Welm: A well-read pre-trained language model for chinese. *arXiv preprint arXiv:2209.10372*, 2022.

J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

R. Sutton. The bitter lesson, 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

Z. Talat, A. Névéol, S. Biderman, M. Clinciu, M. Dey, S. Longpre, S. Luccioni, M. Masoud, M. Mitchell, D. Radev, S. Sharma, A. Subramonian, J. Tae, S. Tan, D. Tunuguntla, and O. van der Wal. You reap what you sow: On the challenges of bias evaluation under multilingual settings. In *Challenges & Perspectives in Creating Large Language Models*, 2022. URL https://openreview.net/forum?id=rK-7NhfSIW5.

Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, and D. Metzler. Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations*, 2021.

Y. Tay, J. Wei, H. W. Chung, V. Q. Tran, D. R. So, S. Shakeri, X. Garcia, H. S. Zheng, J. Rao, A. Chowdhery, et al. Transcending scaling laws with 0.1% extra compute. *arXiv preprint arXiv:2210.11399*, 2022.

R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

T. H. Trinh and Q. V. Le. A simple method for commonsense reasoning, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

P. Villalobos, J. Sevilla, L. Heim, T. Besiroglu, M. Hobbhahn, and A. Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*, 2022.

A. Virtanen, J. Kanerva, R. Ilo, J. Luoma, J. Luotolahti, T. Salakoski, F. Ginter, and S. Pyysalo. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*, 2019.

Q.-H. Vuong, V.-P. La, T.-H. T. Nguyen, M.-H. Nguyen, T.-T. Le, and M.-T. Ho. An ai-enabled approach in analyzing media data: An example from data on covid-19 news coverage in vietnam. *Data*, 6(7), 2021. ISSN 2306-5729. doi: 10.3390/data6070070. URL https://www.mdpi.com/2306-5729/6/7/70.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://www.aclweb.org/anthology/W18-5446.

A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537, 2019b. URL http://arxiv.org/abs/1905.00537.

B. Wang and A. Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

C. Wang, K. Cho, and J. Gu. Neural machine translation with byte-level subwords. *CoRR*, abs/1909.03341, 2019c. URL http://arxiv.org/abs/1909.03341.

S. Wang and P. Kanwar. Bfloat16: The secret to high performance on cloud tpus, 2019. URL https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus.

S. Wang, Y. Sun, Y. Xiang, Z. Wu, S. Ding, W. Gong, S. Feng, J. Shang, Y. Zhao, C. Pang, J. Liu, X. Chen, Y. Lu, W. Liu, X. Wang, Y. Bai, Q. Chen, L. Zhao, S. Li, P. Sun, D. Yu, Y. Ma, H. Tian, H. Wu, T. Wu, W. Zeng, G. Li, W. Gao, and H. Wang. Ernie 3.0 titan: Exploring larger-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2112.12731*, 2021.

T. Wang, A. Roberts, D. Hesslow, T. L. Scao, H. W. Chung, I. Beltagy, J. Launay, and C. Raffel. What language model architecture and pretraining objective works best for zero-shot generalization? In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22964–22984. PMLR, 17–23 Jul 2022a. URL https://proceedings.mlr.press/v162/wang22u.html.

Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2022b.

J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners, 2021a.

J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021b.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and É. Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4003–4012, 2020.

J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. Van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

H. A. Wibowo. Recibrew. https://github.com/haryoa/ingredbrew, 2020.

A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1101.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

H. Wu, G. Diamos, J. Wang, S. Cadambi, S. Yalamanchili, and S. Chakradhar. Optimizing data warehousing applications for GPUs using kernel fusion/fission. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, pages 2433–2442, 2012. doi: 10.1109/IPDPSW.2012.300.

M. Xia, M. Artetxe, C. Zhou, X. V. Lin, R. Pasunuru, D. Chen, L. Zettlemoyer, and V. Stoyanov. Training trajectories of language models across scales. *arXiv preprint arXiv:2212.09803*, 2022.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *CoRR*, abs/2010.11934, 2020a. URL https://arxiv.org/abs/2010.11934.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020b.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. XLnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 2019.

P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. doi: 10.1162/tacl_a_00166. URL https://aclanthology.org/Q14-1006.

J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, B. Hutchinson, W. Han, Z. Parekh, X. Li, H. Zhang, J. Baldridge, and Y. Wu. Scaling autoregressive models for content-rich text-to-image generation, 2022.

S. Yuan, H. Zhao, Z. Du, M. Ding, X. Liu, Y. Cen, X. Zou, Z. Yang, and J. Tang. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68, 2021. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.06.001. URL https://www.sciencedirect.com/science/article/pii/S2666651021000152.

A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang, C. Li, Z. Gong, Y. Yao, X. Huang, J. Wang, J. Yu, Q. Guo, Y. Yu, Y. Zhang, J. Wang, H. Tao, D. Yan, Z. Yi, F. Peng, F. Jiang, H. Zhang, L. Deng, Y. Zhang, Z. Lin, C. Zhang, S. Zhang, M. Guo, S. Gu, G. Fan, Y. Wang, X. Jin, Q. Liu, and Y. Tian. PanGu-α: Large-scale autoregressive pretrained Chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369*, 2021.

T. Zerrouki and A. Balla. Tashkeela: Novel corpus of arabic vocalized texts, data for auto-diacritization systems. *Data in brief*, 11:147, 2017.

B. Zhang, P. Williams, I. Titov, and R. Sennrich. Improving massively multilingual neural machine translation and zero-shot translation. *arXiv preprint arXiv:2004.11867*, 2020a.

C. Zhang, D. Ippolito, K. Lee, M. Jagielski, F. Tramèr, and N. Carlini. Counterfactual memorization in neural language models. *arXiv preprint arXiv:2112.12938*, 2021.

S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. V. Durme. Record: Bridging the gap between human and machine commonsense reading comprehension, 2018.

S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi. Revisiting few-sample bert fine-tuning, 2020b.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

M. Ziemski, M. Junczys-Dowmunt, and B. Pouliquen. The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3530–3534, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1561.

B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, and W. Fedus. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.