GRANITE 3.0 LANGUAGE MODELS

Granite Team, IBM[1]
[1]See Contributions and Acknowledgments section for full author list.
Please send correspondence to granite-inquiries@ibm.com.

ABSTRACT

This report presents Granite 3.0, a new set of lightweight, state-of-the-art, open foundation models ranging in scale from 400 million to 8 billion active parameters. Equipped with native support of multilingual, coding, function calling, and strong safety performance, these models target enterprise use cases, including on-premise and on-device settings. Evaluations on a comprehensive set of tasks demonstrate that our models consistently reach state-of-the-art performance for their size (as shown in Figure 1 and 2). This report also discloses technical details of pre-training and post-training that may help the research community accelerate the collective efforts to develop open foundation models. We publicly release pre-trained and post-trained versions of all our Granite 3.0 models under a standard permissive Apache 2.0 license allowing both research and commercial use. With support from the open source community, the Granite 3.0 models have been integrated with a range of existing tools for quantization, fine-tuning, and deployment.

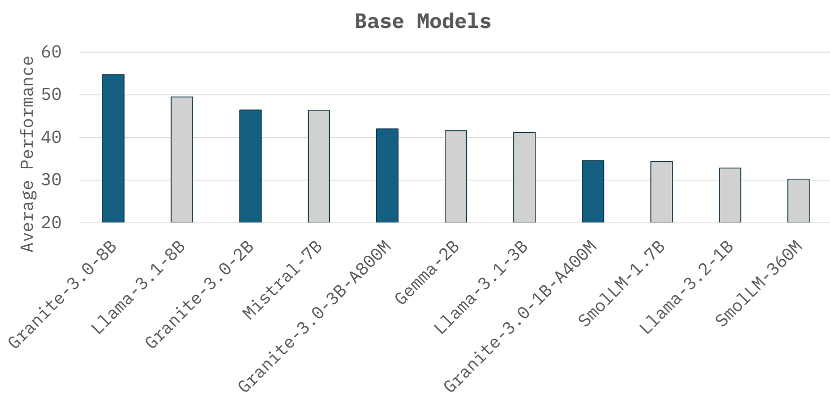


Figure 1: Average performance of base models across 19 tasks from 6 domains.

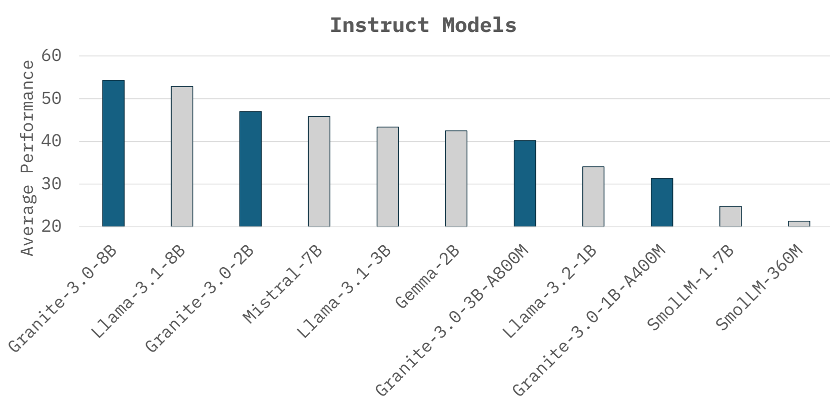
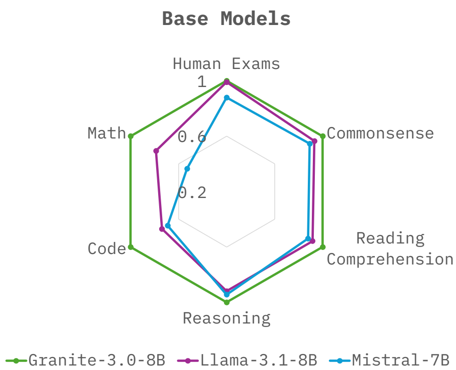
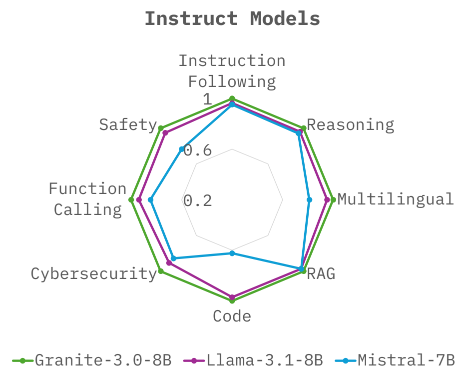


Figure 2: Average performance of instruct models across 23 tasks from 8 domains.



(a)



(b)

Figure 3: The relative performance of Granite-3.0-8B and baseline models across different domains. See Table 8 and Table 9 for details of benchmarks included in each category.

1 Introduction

The adoption of large language models (LLMs) across different applications has spread quickly. While commercial options that are consumer-facing via a web interface or API call are widely available, there is a demand for on-premise models. For accessibility, being able to fine-tune a pretrained LLM for on-premise use requires models with lower hardware requirements.

There are many lightweight models like Gemma (Team et al., 2024) and Llama (Dubey et al., 2024) that perform well and fit the bill. However, in an enterprise setting, the adoption of LLMs can have further constraints. The provenance and transparency around data usage and processing can have legal and compliance implications. In particular, the license that an LLM is released under can also restrict companies from using a model on their specific use cases.

In this report, we present the **Granite 3.0** family of language models natively supporting multilinguality, coding, reasoning, and tool usage, including the potential to be run on constrained compute resources. All the models are publicly released under an Apache 2.0 license for both research and commercial use. The models' data curation and training procedures were designed for enterprise usage and customization in mind, with a process that evaluates datasets for governance, risk and compliance (GRC) criteria, in addition to IBM's standard data clearance process and document quality checks. Specifically, Granite 3.0 includes 4 different models of varying sizes:

- **Dense Models:** 2B and 8B parameter models, trained on 12 trillion tokens in total.
- **Mixture-of-Expert (MoE) Models:** Sparse 1B and 3B MoE models, with 400M and 800M activated parameters respectively, trained on 10 trillion tokens in total.

Accordingly, these models provide a range of options with different compute requirements to choose from, with appropriate trade-offs with their performance on downstream tasks. At each scale, we release a base model — checkpoints of models after pretraining, as well as instruct checkpoints — models finetuned for dialogue, instruction-following, helpfulness, and safety. The base models are trained from scratch with a two-stage training procedure. In stage 1, our dense and MoE models are trained on 10 trillion and 8 trillion tokens, respectively. Stage 1 training data consists of unstructured multilingual language data from diverse sources across academia, the internet, enterprise (e.g., financial, legal), and code, including publicly available datasets with permissive licenses. In stage 2, we train on a mixture of 2 trillion tokens of data. Some of the data sources for stage 2 are the same as the stage 1 data sources, mixed with a small amount of high-quality open-source and synthetic corpora with permissive licenses. The data mixtures are derived through a data mixture search focusing on robustness across different domains and tasks. The instruct models are derived by supervised fine-tuning (SFT) of the pre-trained checkpoints, followed by model alignment using reinforcement learning (PPO, BRAIn (Pandey et al., 2024)). We find that both SFT and PPO/BRAIn are important for improved performance on downstream automatic evaluations, including better chat capabilities.

Additionally, the models were trained with techniques that leverage different methods found in the existing literature: μP (Yang & Hu, 2020; Yang et al., 2022; 2023) allowed for hyperparameter transfer after a hyperparameter search on smaller models, and Power scheduler (Shen et al., 2024c) allowed for learning rate transfer across batch size and total number of training tokens. For our MoE models, we used a dropless MoE (Gale et al., 2023) approach for better model performance using the ScatterMoE (Tan et al., 2024) implementation.

Experiment results show that our Granite 3.0 models outperform models of similar parameter sizes on many benchmarks, demonstrating strong performance in knowledge, reasoning, function calling, multilingual, code support, as well as enterprise tasks like cybersecurity and retrieval augmented generation (RAG). Figure 3 shows that our Granite-3.0-8B models consistently outperform Llama-3.1-8B and Mistral-7B on various domains. The key advantages of Granite 3.0 models are:

- **Lightweight:** Our largest dense model has 8 billion parameters, and our smallest MoE model has an activated parameter count of 400 million, enabling hosting, or even fine-tuning, on more limited compute resources.
- **Robust Models with Permissive License:** Combined with excellent performance across various benchmarks, our Granite 3.0 models provide a great foundation for enterprise customization. All our models, including instruct variants, use an Apache 2.0 license, allowing for more consumer and enterprise usage flexibility over the more restrictive licenses of other available models in the same class.
- **Trustworthy Enterprise-Grade LLM**: All our models are trained on license-permissible data collected following IBM's AI Ethics principles[1] for trustworthy enterprise usage. We describe in great detail the sources of our data, data processing pipeline, and data mixture search to strengthen trust in our models for mission-critical and regulated applications.

We describe the model architecture and background on MoE models in Section 2. Then, we describe our data collection, filtering, and preprocessing pipeline in Section 3. We then go into detail about our data mixture and hyperparameter search for pretraining in Section 4, followed by our post-training methodology in Section 5, and our compute infrastructure in Section 6. Section 7 describes the results of our comprehensive evaluation of the trained models, including a comparison with other open-source LLMs. Finally, Section 8 discusses the social harms and risks of this project.

2 Model Architecture

The Granite 3.0 language models are based on two architectures: a decoder-only dense transformer and a decoder-only sparse Mixture-of-Expert (MoE) transformer.

Table 1: Hyperparameters for Granite 3.0 models.

Model	2B	8B	1B-A400M	3B-A800M
Embedding size	2048	4096	1024	1536
Number of layers	40	40	24	32
Attention head size	64	128	64	64
Number of attention heads	32	32	16	24
Number of KV heads	8	8	8	8
MLP hidden size	8192	12800	512	512
MLP activation	SwiGLU	SwiGLU	SwiGLU	SwiGLU
Number of Experts	–	–	32	40
MoE TopK	–	–	8	8
Initialization std	0.1	0.1	0.1	0.1
Sequence Length	4096	4096	4096	4096
Position Embedding	RoPE	RoPE	RoPE	RoPE
#Parameters	2.5B	8.1B	1.3B	3.3B
#Active Parameters	2.5B	8.1B	400M	800M
#Training tokens	12T	12T	10T	10T

[1] https://www.ibm.com/impact/ai-ethics

2.1 Dense Models

Granite 3.0 2B and 8B dense models share a similar architecture as popular language models like Llama and our previous Granite Code models Mishra et al. (2024), ensuring strong compatibility with open-source inference and fine-tuning pipelines. We use Grouped Query Attention (GQA; Ainslie et al. 2023) with 8 key-value heads to get a good balance between memory cost and model performance, and Rotary Position Embedding (RoPE; Su et al. 2024) to model the relative position between tokens. For the MLP layers, Granite 3.0 Dense models use SwiGLU as the activation function. Before each MLP and attention layer, we use RMSNorm to normalize the layer's input. We also share parameters between the input embedding and the output linear transform. This reduces the size of the model, and we have observed that the tying of these embeddings have zero, or even a positive impact on model performance.

2.2 Mixture-of-Expert models

Granite 3.0 1B and 3B MoE models use similar architecture as Granite Dense models, with the MLP layers substituted with MoE layers. A Mixture of Experts (MoE) layer comprises N modules $f_1, \ldots, f_N$ and a router $g(e \mid \mathbf{x})$. Given an input $\mathbf{x}$ to the MoE layer, the router predicts a probability distribution over the N modules. Of these, we select the top k experts. When $k < N$, we are using a Sparse Mixture of Experts (SMoE; Shazeer et al. 2017). For this series of Granite MoE models, we use a linear layer to model the router:

$$\mathbf{s} = \mathbf{W}_{\mathrm{router}}\mathbf{x}, \tag{1}$$

$$g(e \mid \mathbf{x}) = \begin{cases} \mathrm{softmax}\left(\mathrm{Top}k\left(\mathbf{s}\right)\right)_i, & \mathbf{s}_i \in \mathrm{Top}k\left(\mathbf{s}\right) \\ 0, & \mathbf{s}_i \notin \mathrm{Top}k\left(\mathbf{s}\right) \end{cases} \tag{2}$$

where $\mathbf{W}_{\mathrm{router}}$ is the expert embedding matrix of shape (N, D_{emb}), and Top$k$ is the operator that selects the top k logits from $\mathbf{s}$. The final output of the SMoE is then given by

$$y = \sum_{e=1}^{N} g(e \mid \mathbf{x}) \cdot f_e(\mathbf{x}) \tag{3}$$

When $g(e \mid \mathbf{x}) = 0$, $f_e(\mathbf{x})$ will not need to be evaluated, thus reducing computation cost during training and inference. The key designs of the Granite MoE models are summarized below:

Dropless Token Routing. Since each token selects experts independently, some experts could receive more tokens than others. In previous MoE models, like Switch Transformer (Fedus et al., 2022) and Deepseek-V2 (Liu et al., 2024a), a capacity cap is set for each expert or device, and the extra tokens that exceed the cap are dropped. As observed in Gale et al. (2023), this cap negatively affects the model training stability and loss. In our training, we use ScatterMoE (Tan et al., 2024), a dropless MoE implementation, to avoid token dropping and improve training efficiency.

Fine-grained Experts. Recent studies (Krajewski et al., 2024; Dai et al., 2024) suggest that setting the size of experts in MoE to mirror the feed-forward layer is not optimal. Instead, increasing the expert granularity, number of experts, and number of activated experts could increase the possible combinations of experts and result in better model performance. Following these observations, we use fine-grained experts and a larger number of activated experts in Granite 3.0 MoE models. Specifically, we use a top-k of 8 out of 32 and 40 experts respectively for the 1B and 3B MoE models.

Load Balancing Loss. To avoid routing tokens repeatedly to the same expert and wasting the extra capacity in other experts, we use the frequency-based auxiliary loss introduced in Fedus et al. (2022)

$$\mathcal{L}_b = N \sum_{i=1}^{N} f_i P_i \tag{4}$$

where N is the number of experts, f_i is the fraction of tokens dispatched to expert i, and P_i is the fraction of the router probability allocated for expert i. Intuitively, this loss penalises over-usage of experts, thus 'balancing' the load. To improve the training stability, we also use the router z-loss introduced in Zoph et al. (2022):

$$\mathcal{L}_z = \frac{1}{B} \sum_{i=1}^{B} \left(\log \sum_{j=1}^{N} \exp(x_j^i) \right)^2 \tag{5}$$

where B is the number of tokens, x is the logits given by router. This loss penalises logits of the router when it has extreme values, allowing the router to adapt better during training in order to better assign experts. The final loss is the weighted sum of language model loss and two auxiliary losses.

3 Training Data

Granite 3.0 language models are trained using data from various sources such as unstructured natural language text and code data from the Web curated by IBM, a collection of synthetic datasets generated by IBM, and publicly available high-quality datasets with permissible licenses. For governance, all our data undergoes a data clearance process subject to technical, business, and governance review. This comprehensive process captures critical information about the data, including but not limited to their content description, ownership, intended use, data classification, licensing information, usage restrictions, how the data will be acquired, as well as an assessment of sensitive information (i.e, personal information). For code, we annotate each code file with license information associated with the respective repository, found via Github APIs and only keep files with permissive licenses for model training. In addition, we also filter out all data obtained from sources that match URLs in IBM's URLs blocking-list. Below, we provide a brief overview about our data processing steps and refer to the Appendix B.1 for details on individual datasets used in different stages of model training.

3.1 Curated Web Data

We curate a massive collection of unstructured data from the Web, obtained from academic, enterprise (e.g., financial, legal, biomedical), code, and other publicly available sources, e.g., open-source datasets like FineWeb (Penedo et al., 2023), DCLM (Li et al., 2024a) for language and Github Code Clean[2], StarCoderdata[3] for code domain. To ensure the quality of our curated web data, we have developed a comprehensive data preprocessing procedure, as follows.

Text Extraction. Text extraction is the first step in the processing of unstructured data crawled from web, and is used to extract text from various documents into a standardized format for further processing. After text extraction, we adopt fasText [4] for language identification at a document level to detect the dominant language. With language identification, we specifically select documents annotated with 12 languages namely English, German, Spanish, French, Japanese, Portuguese, Arabic, Czech, Italian, Korean, Dutch, or Chinese as the dominant language. While our training data consists of primarily-English data, we utilize high-quality documents from these other eleven languages to improve multilinguality of Granite 3.0 models.

Deduplication. Deduplication aims to identify and remove duplicate documents to improve overall quality of a dataset. We adopt an aggressive deduplication strategy including both exact and fuzzy deduplication to remove documents having (near) identical content. For exact deduplication, we first compute SHA256 hash on the document content and remove records having identical hashes. Then, we follow a two-step fuzzy deduplication strategy: (1) computing MinHashes of all the documents and then utilize Locally Sensitive Hashing (LSH) to group them based on their MinHash fingerprints, and (2) measuring Jaccard similarity between each pair of documents in the same bucket to annotate documents except one as duplicates based on a similarity threshold. We apply this near-deduplication process to both language and code data to enhance the richness and diversity of the training dataset.

[2] https://huggingface.co/datasets/codeparrot/github-code-clean
[3] https://huggingface.co/datasets/bigcode/starcoderdata
[4] https://huggingface.co/facebook/fasttext-language-identification

HAP and Malware Filtering. To reduce the likelihood of generating hateful, abusive, or profane (HAP) language from the models, we make diligent efforts to filter HAP content from our training set. We compute HAP scores based on a HAP detector trained by IBM [5] at the sentence level and filter out documents which exceeds a certain threshold. For code documents, we first create a dictionary of HAP keywords and then annotate each document with the number of occurrences of such keywords in the content including comments. We filter out documents which exceeds the HAP threshold, computed based on a distributional analysis as well as manual inspection of code files. Please refer to Mishra et al. (2024) for more details on the processing of code data. We also scan our datasets using ClamAV[6] to identify and remove instances of malware, especially in the source codes.

Document Quality Filtering. Quality annotation aims to identify documents with low linguistic value using both heuristics and a classifier. Specifically, we follow Gopher quality filtering criteria (Rae et al., 2021) to remove low quality documents that contain for example bullet points ratio of greater than 90%, ellipsis line ratio of greater than 30% and symbol to word ratio of greater than 10%, etc. Besides heuristics, we also adopt a classifier-based filtering that assigns a perplexity score using the KenLM linear classifier [7], pre-trained on a small collection of known high quality documents (e.g., Wikipedia articles). For any document, the KenLM linear classifier provides a score of the document's similarity to a training corpus, indicating overall quality for model training.

We also apply several heuristics to filter out lower-quality code (Mishra et al., 2024): (1) remove files with fewer than 25% alphabetic characters, (2) filter out files where the string "`<?xml version=`" appears within the first 100 characters, (3) for HTML files, only keep files where the visible text makes up at least 20% of the HTML code and has a minimum length of 100 characters, (4) for JSON and YAML files, only keep files that have a character count ranging from 50 to 5000 characters.

3.2 Synthetic Data

Existing permissive datasets are becoming increasingly inadequate for training models with specific capabilities, e.g., coding, reasoning, and safety etc during post-training. While collecting high-quality data from humans is a potential solution for this, it is a time-consuming and costly endeavor. To address this challenge, we conduct an in-depth exploration of synthetic data generation (SDG) as an alternative for Granite models. Recently, the introduction of synthetic data pipelines such as, Self-Instruct (Wang et al., 2023), Evol-Instruct (Xu et al., 2023), and MagPie (Xu et al., 2024) leveraged ways to synthetically produce datasets nearly as comprehensive, competitive, and diverse as those created by humans. Inspired by these recent methods, our synthetic datasets are composed of input-output pairs, cover single and multi-turn scenarios, and target a generic or specific nature. In this section, we provide an overview about the synthetic datasets, what data domains they contribute to, and how we filter them for effective post-training of our models.

Generic Instruction Data. We build generic instruction data in form of instruction-response pairs primarily using Evol-Instruct (Xu et al., 2023) and MagPie (Xu et al., 2024) methods. Specifically Evol-Instruct takes an initial seed of instruction data to generate improved complex versions of them by randomly selecting in-depth or in-breadth evolving prompt templates. We verify the quality of evolved instructions through a set of heuristics (e.g., character length, word count, seed instruction leakage) and exclude instructions that do not pass this quality verification from the final dataset. We repeat this evolution 5 times, and at each iteration, a mix of either original seed instructions or evolved instructions (that passed the quality check) are used as input for the following iteration. A list of synthetic datasets generated using Evol-Instruct can be found in Appendix B.2.

Unlike Evol-Instruct, MagPie generates high-quality instruction data without relying on prompt engineering or seed questions. Instead, it directly constructs instruction data by prompting a LLM with a pre-query template for sampling instructions. Following (Xu et al., 2024), we use 12 combinations of temperature and top-p parameters by prompting a couple of teacher models with their respective pre-queries followed by greedy sampling to generate the corresponding responses. Our final unfiltered samples target varied between 5M and 10M depending on the LLM used for generation. In addition, we also extend filtered versions of our single-turn datasets to create multi-turn datasets to enhance

[5] https://huggingface.co/collections/ibm-granite/granite-guardian-66db06b1202a56cf7b079562
[6] https://www.clamav.net/
[7] https://huggingface.co/edugp/kenlm

chat capabilities. We only use open-source teacher models with a non-restrictive license to generate instructions and their respective responses. A list of synthetic datasets generated using Evol-Instruct and MagPie can be found in Appendix B.2.

Code. As demand for software development surges, it is more critical than ever to increase software development productivity, and LLMs provide promising path for augmenting human programmers. To improve coding capabilities of Granite 3.0 models, we focus on generating high quality synthetic code data and creating quality filters to remove bad samples from our training data. Specifically, inspired by Starcoder2-Instruct[8], we extend the OSS self-instruct pipeline to 6 coding languages: JavaScript, TypeScript, C, C++, Go, and Python, and use filtered pretraining data as the seed data with `granite-34b-code-instruct` as the teacher model for generating instruction-response pairs. We generate 235,000 samples with this method, which serves as one of the major code generation dataset in our supervised finetuning. Beyond code generation, we also generate samples to cover tasks like code explanation, docstring, and pseudocode generation by using a generate, backtranslate, and filter process, as in (Dubey et al., 2024). Specifically, we prompt a teacher model to generate code explanation, docstring, and pseudocode from the seed functions. These tasks are intentionally grouped together so the model's reasoning can improve with the combination of tasks. For filtering, we prompt the model to grade the original seed function against the synthetic code for both fidelity and complexity, and we keep samples that only pass a predefined threshold. We also adopt a similar pipeline for generation synthetic data for tasks like unit test generation and code debugging, helping our models to perform better on code fixing and explanation.

We also leverage SDG to collect multi-turn data, by incorporating execution output as feedback along with human feedback. Following (Zheng et al., 2024), we generate 50k multi-turn dialogues with code execution and synthetic code reviews. We identify python samples via model classification and heuristics in two open source datasets, namely Glaive Code Assistant [9] and Code Instructions Alpaca [10]. We filter the instructions by complexity using LLMaJ with a complexity scale of 1-5. We then use the instruction from the filtered dataset as the initial prompt in an mulit-turn agentic pipeline that consists of agents for writing, executing and reviewing code built with the AutoGen framework. Dialogues that produce trajectories with executable code with passing unit tests after a round of synthetic code review and update are included in the final dataset.

Reasoning. Reasoning with LLMs has been in the forefront (Plaat et al., 2024; Zhang et al., 2024a), given both the evolution of benchmarks and the inconclusive discussions around the reasoning abilities of LLMs (Kojima et al., 2023; Mirzadeh et al., 2024). However, the progress with chain-of-thought and evolution of Agentic behaviors has shifted focus towards innovative mechanisms to generate reasoning and planning traces with human validation of each step (et al., 2024; Lightman et al., 2023). We employ two main techniques for generating synthetic reasoning data:

- Code-Assisted Synthetic Data Generation: We use code-assisted SDG for primarily algorithmic tasks (Li et al., 2023b). Within each reasoning category, we use domain-specific seed prompts and seed data to introduce diversity. Python code execution is used both to generate and validate chain-of-thought and the final answer for various reasoning tasks.
- Knowledge-based Data Generation: We utilize multiple knowledge graphs such as ATOMIC (Hwang et al., 2021) and Wikidata (Vrandečić & Krötzsch, 2014) to generate multi-hop reasoning data with the chain-of-thought that includes commonsense and encyclopedia knowledge. The initial knowledge graph seed structures are a combination of template-based and random-walk-based techniques, enabling the grounding of the reasoning traces being generated. Grounding helps in eliminating incorrect reasoning traces with both right and wrong final answers.

Retrieval Augmented Generation (RAG) RAG is widely recognized as a promising approach to address common challenges in large language models, such as factual inaccuracies, outdated knowledge, and limitations in domain-specific expertise (Chen et al., 2024a). To generate synthetic data for improving RAG capabilities, following (Lee et al., 2024b), we first input a document and prompt an LLM to generate a user question. These questions span various types commonly found

[8] `https://huggingface.co/blog/sc2-instruct`
[9] `https://huggingface.co/datasets/glaiveai/glaive-code-assistant-v3`
[10] `https://huggingface.co/datasets/TokenBender/code_instructions_122k_alpaca_style`

in information-seeking tasks, and to ensure that the generated questions align with the designated question types, we incorporate question-type-specific CoT prompts, to guide the language model in reasoning through the grounding document (Lee et al., 2024b). We then employ a retriever (ELSER with `sentence-transformers/all-MiniLM-L6-v2` sentence embeddings) to dynamically select the top-k relevant passages from a pre-constructed document index, extending the user questions generated in the first step into multi-turn, multi-document grounded conversations. The retrieved passages, along with the query, are provided to the LLM, which is then prompted to generate a response. For subsequent turns, we first prompt the LLM to generate a query given dialog history and retrieved passages and then generate the answer as described above. Furthermore, an LLM-as-a-Judge module is used to filter out dialogues with incorrect responses by evaluating all context-response pairs within each dialogue. A larger language model assesses the accuracy of responses based on dialogue history and the current query, and any dialogues containing incorrect question-answer pairs are discarded. The multi-turn RAG data generation use two pblicly available datasets, QuAC (Choi et al., 2018) and MultiDoc2Dial (Feng et al., 2021) as seed datasets.

Tool Use. Tool (or function) calling is now considered one of the fundamental capabilities that LLMs need to possess (Abdelaziz et al., 2024). In real-world applications, tool invocation can vary in complexity, ranging from a single turn with a single tool (the simplest case) to more challenging multi-turn interactions involving multiple sequential/nested tool calls. To train LLMs to utilize tools requires a broad spectrum of training data covering different tool calling scopes.

In the recent past, there has been extensive research on transforming existing curated and manually annotated datasets into a function calling format. While we leverage curated datasets such as API-BLEND (Basu et al., 2024) and APIGen (Chen et al., 2024b), these datasets do not address the evolving landscape of function calling benchmarks such as multi-turn tool calls, nested tool calls, and tool relevance, etc. To address this, we extend existing SDG techniques to introduce new features of function calling in LLMs. These pipelines also include generating data that calls APIs via different programming languages (Guo et al., 2024).

Cybersecurity. To create a comprehensive and diverse dataset capable of supporting instruction tuning for various security-related tasks, we adopt the process in Levi et al. (2024), which consists of two main steps, as follows. In the first generation step, we concentrate on producing high-quality instructions derived from predefined schemas. These schemas are formulated through expert-driven analysis of a diverse set of security datasets, examining the relationships between different entities across datasets. This approach ensures that the instructions capture the nuances of various security concepts and tasks. More specifically, each predefined schema has rules that dictate how the data source should be processed into instructions, using parsers developed specifically for these security data sources. This guarantees that the generated instructions focus on the important and unique characteristics of the data source and are representative of real-world security scenarios. In the second generation step, the diversity and complexity of the initial generated dataset is expanded by employing a hybrid synthetic content-grounded data generation process. Specifically, we combine Evol-Instruct (Xu et al., 2023) and Self-Instruct (Wang et al., 2022) alongside content-grounded generation and evaluation pipelines. Additionally, we implement a routing mechanism between the two generation methods to help reduce hallucinations.

This process leverages the initial set of instructions and data from the first generation step to generate additional instructions that follow the established schemas while increasing the model's overall generalizability. By incorporating content-grounded synthetic data, we increase the diversity and volume of the final dataset, ultimately leading to more robust and capable security models.

We leverage various publicly available security data sources, namely MITRE ATT&CK[11], CWE[12], CVE[13], CAPEC[14], Security Wikipedia, Security interview Q&A, Threat reports, BRON (Hemberg et al., 2020), SIEM alert rules, Sigma rules[15], and Security Stack Exchange to generate both rules-based and synthetic security instructions. Our final synthetic security dataset consists of various

[11] https://attack.mitre.org
[12] https://cwe.mitre.org
[13] https://cve.mitre.org
[14] https://capec.mitre.org
[15] https://github.com/SigmaHQ/sigma

instruction types, such as open/closed book question answering, yes/no questions, multi-choice Q&A, CoT, logic validation, odd/leave one out multi-choice Q&A, question generation, query/rule explanation and generation, TTP mapping, and others.

Multilingual. To improve our model's machine translation quality, we include parallel text from datasets such as ParaCrawl [16], WikiMatrix (Schwenk et al., 2019a), and NLLB/CCMatrix (Schwenk et al., 2019b). We apply extensive filtering based on language-specific heuristics and model-based scoring to select only the highest-quality translation pairs from these datasets. For example, filtering heuristics include steps such as removing samples that do not have a minimum amount of words or where source and target have too many tokens in common, contain too many repeated characters or tokens, or contain too many UTF-8 control characters. We also all apply more language-specific filtering heuristics, e.g., for CJK languages or Arabic, we set a minimum threshold for the percent of characters in the target language script. Model-based filtering comprises language id and calculating alignment scores via a multilingual sentence embedding model (Artetxe & Schwenk, 2019).

Moreover, to enhance multilingual capabilities in multi-turn conversation scenarios, we translate a subset of the publicly available Daring Anteater [17] SFT dataset into our targeted languages. For translation, we use an existing LLM to translate each turn separately, but make sure that text formatting and especially code blocks within each turn are preserved during translation.

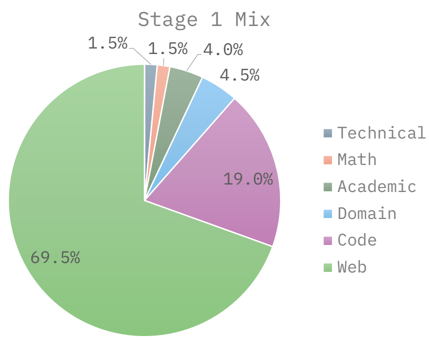
Safety. To safeguard our models, we leverage synthetic data as a powerful source for augmenting AI safety training and aligning models in a targeted way. An AI risk taxonomy is a repository that classifies and structures categories of risk. IBM Research uses its AI risk taxonomy as part of a broader set of safety measures that apply to its development of foundation models. The taxonomy helps to categorize known risks, which are used to generate synthetic data, for the purpose of aligning language models. We leverage the following taxonomy for SDG that covers 7 high-level categories:

- Malicious Use: illegal activities, unethical or unsafe actions, and violence and extremism.
- System Risks: security and operational risks.
- Information Hazards: sensitive and personal information.
- Discrimination: a wide range of discrimination, including implicit and explicit bias.
- Societal Risks: disinformation, propaganda, and voter suppression.
- Human-Chatbot Interactions: mental health, child harm, and self-harm.
- Multi-Modal Requests: various forms of undesirable requests related to multi-modal support.

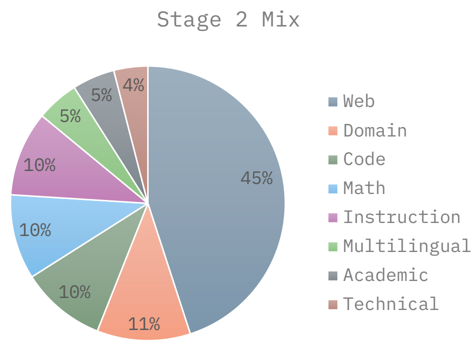
Our safety taxonomy has been informed by internal research as well as opensource AI risk taxonomies research conducted by MIT and MLCommons [18]. A diverse set of people distributed across different IBM locations worldwide with diverse expertise and socio-cultural background contributed quality seeds to address inappropriate prompts. Additionally, research into adversarial attacks also informed the seeds gathered to safeguard our model against jailbreaks and prompt injection attacks. Specifically, our synthetic safety data includes prompt-response pairs across a broad range of scenarios, covering direct requests across a safety taxonomy with direct questions, comparative questions, hypothetical prompts, adversarial attacks, and multi-turn interactions designed to expose unsafe behavior. Once generated, quality and consistency checks of the synthetic data are also applied which includes both extensive automated and manual reviews. An iterative approach was also used to improve the safety alignment data by analyzing the resulting model's behavior and producing more synthetic data to increase the safety coverage as needed.

Quality Filtering. When using synthetic data, we run several stages of filtering over it, removing samples that are very short, easy for a reference model, unclear instructions or duplicated samples. We follow (Xu et al., 2024) and leverage (1) LLM-as-judge (Zheng et al., 2023) to determine the category, quality, and difficulty of instructions, and (2) the computation of minimum neighbor distance in the

[16] http://paracrawl.eu
[17] https://huggingface.co/datasets/nvidia/Daring-Anteater
[18] https://mlcommons.org/



(a)



(b)

Figure 4: Data mixture for pretraining stages. The percentage for individual datasets has been merged into different categories. Best viewed in color.

embedding space to identify near duplicates (Douze et al., 2024). Our sample annotation pipeline uses `Mistral-7B-Instruct-v0.3` [19] as the reference model (judge) and consists of the following steps:

- Instructions Annotation: We annotate the generated instructions with category (creative writing, advice seeking, planning, and math, etc), difficulty ('very easy', 'easy', 'medium', 'hard', or 'very hard'), and instruction quality ('very poor', 'poor', 'average', 'good', and 'excellent').
- Responses Annotation: We asses the quality of responses conditioned on their respective instruction using an LLM as judge (in a scale of 1-5).
- Sample-level Annotation: Particularly important for multi-turn datasets, we prompt an LLM for assessing overall quality of a conversation between a user and an assistant (including all turns).
- Duplicates Annotation: We measure the similarity using minimum neighbor distance in the embedding space (Xu et al., 2024). For multi-turn data, however, we concatenate the inputs from all turns and compute minimum neighbor distance in the embedding space using the full conversations.

4 Pre-training

Granite 3.0 language models are trained on 10T to 12T tokens of language and code data, sourced from different domains. Data is tokenized via byte pair encoding (BPE, (Sennrich et al., 2015)), employing the same tokenizer as StarCoder (Li et al., 2023d). In this section, we provide details on our two stage training, data mixture and power scheduler used in pretraining the models.

4.1 Data Mixture

Beyond training data quality, the data mixture is another important aspect of model performance. We craft the pretraining data mixture with two goals: 1) maximize the model's performance across a diverse set of domains and tasks without bias toward a specific type of data or task; 2) leverage both high-quality and medium-quality data for optimal performance. To achieve these two goals, we adopt the 2-stage data mixture strategy used in MiniCPM (Hu et al., 2024) and JetMoE (Shen et al., 2024b). In stage 1, we pre-train the model on a large quantity of medium-quality data to learn and memorize the knowledge from diverse domains. In stage 2, we continue pre-training the model on a smaller set of high-quality data mixed with medium-quality data to encourage the model to mimic the behavior of high-quality data and improve the model's performance on downstream tasks.

[19] `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`

4.1.1 Stage 1 Data Mixture

The stage 1 data mixture search focuses on achieving robust language model performance across different domains D_i. Inspired by distributionally robust language model (Oren et al., 2019) and DoReMi (Xie et al., 2024), our target was to minimize the weighted sum of relative domain losses, with respect to a baseline:

$$\min_{\theta} L(\theta, \alpha) := \sum_{i=1}^{|D|} \alpha_i \cdot \left[\frac{1}{\sum_{x \in D_i} |x|} \sum_{x \in D_i} (\ell_\theta(x) - \ell_{\text{ref},i}(x)) \right] \tag{6}$$

where θ is the mixture percentage of pre-training data, $\ell_\theta(x)$ is the negative log-likelihoods of the proxy model for data mixture search, $\ell_{\text{ref},i}(x)$ is the reference model for domain D_i, $|x|$ is the number of tokens in an example x, and α is the weights for different domains. Reference models are small language models trained with in-domain data from D_i. Our assumption here is that the loss reflects the difficulty of the target domain. By subtracting the reference model loss, we normalize the proxy model loss by removing the difficulty factor. This way, we can avoid forcing the model to learn difficult domains even if the model cannot improve further in this domain (Oren et al., 2019).

In the data mixture search, we train thousands of small proxy models with randomly sampled data mixtures to find the optimal data mixture. Each proxy model is a small language model with 10M parameters trained on 15B tokens. The ratio of the number of tokens to parameters is approximately the same as our 8B dense model. Oren et al. (2019) and Xie et al. (2024) suggest that the maximum relative domain loss should be minimized to achieve a distributional robust language model. However, in practice, we notice that some domains are more difficult to learn in this multi-domain learning setting due to conflicts between domains or format mismatches. In other words, if we focus on reducing the maximum relative domain loss, the performance of other domains will be sacrificed. To account for that, we minimize the average relative domain loss:

$$\min_{\theta} L(\theta) := \sum_{i=1}^{|D|} \frac{1}{|D|} \cdot \left[\frac{1}{\sum_{x \in D_i} |x|} \sum_{x \in D_i} (\ell_\theta(x) - \ell_{\text{ref,i}}(x)) \right] \tag{7}$$

After running many experiments, we select the proxy model with minimum $L(\theta)$ and use its data mixture as our stage 1 data mixture. Figure 4(a) shows the data mixture of stage 1.

4.1.2 Stage 2 Data Mixture

The stage 2 data mixture search focuses on improving model performance on a diverse set of downstream tasks from natural language, code, and math domains. Similar to the stage 1, we maximize the weighted sum of performance on the three domains:

$$\max_{\theta} L(\theta, \alpha) := \sum_{i=1}^{|D|} \alpha_i \cdot \left[\frac{1}{|D_i|} \sum_{t \in D_i} \mathrm{Acc}_t(\theta) \right] \tag{8}$$

where Acc_t is proxy model performance on the task t. We use the average across multiple tasks from each domain as the targeted metric to avoid over-fitting on a specific task.

However, in stage 2, we cannot use the small proxy models for the mixture search, because most downstream tasks require a model that is large enough to achieve meaningful performance. Thus, we conduct the mixture search with our 2B dense model. Even with the 2B dense model, we still cannot run thousands of experiments to find the optimal search. Instead, we run a few hundred experiments with 30B tokens from a randomly sampled mixture and use linear regression to fit the correlation between the data mixture and task performance. Figure 5 shows the correlation between the predicted and ground-truth accuracy for different domains. We can see a strong correlation between the two sets of values. Based on the linear regression results, we craft the final stage 2 mixture to achieve robust performance across different domains. Figure 4(b) shows the data mixture of stage 2.

4.2 Training Hyperparameters

In Shen et al. (2024c), we proposed a systematic way of doing a hyperparameter search on a small scale and 0-shot transfer the hyperparameter to a large scale. The core part of this method is maximum update parameterization and a power scheduler.

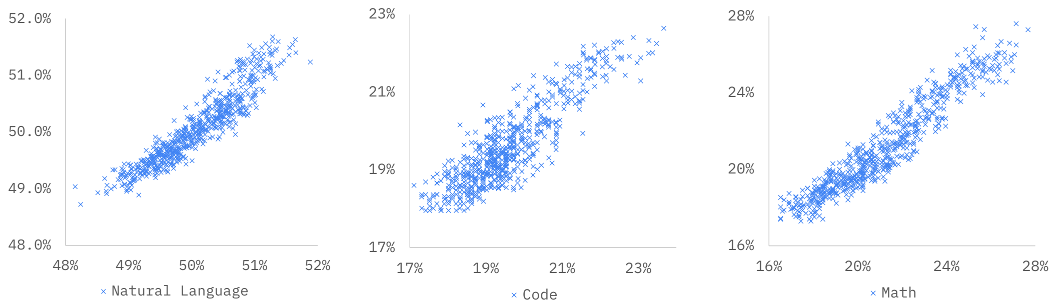


Figure 5: The predicted and ground-truth accuracy for different data mixture samples. The x-axis is the ground truth accuracy, and the y-axis is the predicted accuracy.

Maximal Update Parameterization. (μP) (Yang & Hu, 2020; Yang et al., 2022; 2023) controls initialization, layer-wise learning rates, and activation magnitudes to ensure analytically stable training, independent of a model's width and depth. In addition to improving training stability, μP improves the transferability of training hyperparameters from small proxy models to large models, a technique called μTransfer. The hyperparameter transferability of μP is theoretically justified and empirically demonstrated for width (Yang et al., 2022) and depth (Yang et al., 2023).

Table 2: List of changes applied when using μP. m_{width} is width multiplier, defined as d_m/d_{base}, where d_{base} is the embedding width, d_m is the target model size.

Name	Function
Embedding multiplier	Multiply the embedding output with m_{emb}
Residual Multiplier	Multiply the output of each attention and MLP layer with m_{res} before adding to residual connection
Initialization std	Initialize internal weight matrices (excluding input and output embedding) with standard deviation $\text{init}_{\text{std}}/\sqrt{m_{\text{width}}}$
Learning rate scaling	Set learning rate of internal weight matrices to η/m_{width}
Attention logit scaling	Divide attention logits by d_{head}

We follow the μP config used in CerebrasGPT (Dey et al., 2023) to study the transferability of batch size and learning rate across different numbers of training tokens and model sizes. Table 2 lists the μP changes we applied to model initialization, learning rate, and multipliers.

Power Scheduler. is a new learning rate schedule that includes a linear warmup, a slow power decay, and a fast exponential decay:

$$\text{Power}\,(n) = \begin{cases} \frac{n}{N_{\text{warmup}}} \cdot \eta_{\text{max}} & \text{if } n \leq N_{\text{warmup}} \\ \min\left(\eta_{\text{max}}, \beta a n^b\right) & \text{if } N_{\text{warmup}} < n \leq N - N_{\text{decay}} \\ f(n, N, N_{\text{decay}}) \cdot \beta a \left(N - N_{\text{decay}}\right)^b & \text{if } N - N_{\text{decay}} < n \end{cases} \tag{9}$$

where β is the batch size, n is the number of tokens already trained, a is the amplitude of the learning rate, b is a power-law exponent for decaying the learning with respect to the number of trained tokens, and η_{max} is the learning rate upper bound that rejects very large learning. Shen et al. (2024c) shows two benefits of a power scheduler: 1) it enables zero-shot transferability of learning rate between different batch sizes and numbers of training tokens, such that we can use small batch size and number of training tokens to search for optimal learning rate, then use it for large scale training runs; 2) it doesn't require a predefined number of training tokens or steps, such that we can start the training and do early exit or add additional tokens to the training and still get optimally converged model.

Combining these two methods, we conducted the hyperparameter search on a very small scale (36M parameters, up to 128B tokens) to get the optimal hyperparameters. We then zero-shot transferred this set of hyperparameters to all of our model training. More details about the hyperparameter search can be found in Shen et al. (2024c).

For all our models, we use AdamW optimizer (Diederik, 2014) with $\beta_1 = 0.9, \beta_2 = 0.95$ and weight decay of 0.1 for training all our models. For the learning rate, we use the power scheduler with $a = 4$, $b = -0.51$, and $\eta_{\max} = 0.02$, and an initial linear warmup step of 2500 iterations. We use a batch size of 4M tokens during both stages of pretraining.

4.3 Model Parallelism

We use a combination of 3D parallelism (Tensor Parallelism (Shoeybi et al., 2020), Pipeline Parallelism (Narayanan et al., 2021b) and Data Parallelism (Li et al., 2020)) for training all our models. We use tensor parallelism to split the weight matrices and activations and pipeline parallelism to slice the model along the layers. We put an equal compute load on each pipeline stage except the first and last pipeline stages, which contain the embedding matrix and the LM head, respectively. The 2B dense model is trained with 256 GPUs using tensor parallel sharding on 2 GPUs, and the 8B dense model is trained on 768 GPUs using 4x tensor parallelism and 4x pipeline parallelism. We use the 1F1B (1-Forward 1-Backward) schedule (Narayanan et al., 2021a;b) for efficient pipeline parallelism to reduce the memory consumption for the in-flight microbatches. The 1F1B schedule makes the memory consumption proportional to the pipeline depth instead of the number of in-flight microbatches, which can be quite large when training such models. We shard the optimizer on the Data Parallel process group similar to ZeRO-1 (Rajbhandari et al., 2020) for reducing the optimizer memory footprint during training. Because tensor parallelism is extremely latency sensitive and blocking in nature, we only do tensor parallelism within a server node while pipeline parallelism and data Parallelism can span across nodes. We only use data parallelism without any model parallelism to train the MoE models. They are trained using 128 and 256 GPUs using ZeRO-1 sharding (Rajbhandari et al., 2020) to shard the optimizer across multiple GPUs. To accelerate training, we use FlashAttention 2 (Dao et al., 2022; Dao, 2023), the persistent layernorm kernel, Fused RMSNorm kernel (depending on the model), and the Fused Adam kernel available in NVIDIA's Apex library.

5 Post-training

We develop the post-training (instruct) variants of our Granite 3.0 models by further training the pre-trained checkpoints, focusing on instruction-following capabilities and alignment with human values. We employ a diverse set of techniques with a structured chat format, including curriculum-based supervised finetuning, model alignment using proximal policy optimization (PPO), best-of-N sampling, BRAIn (Pandey et al., 2024), and model merging.

5.1 Structured Chat Template

While a pre-trained model may generate reasonable responses to directives—or, instructions—a standardized chat format is commonly used in post-training. A common format not only reinforces the structure of query and response pairs used in the applications of instruct-styled models but also allows for further control sequences to denote different actors within a single input provided to the model, such as information from external systems. To support both human-AI and machine-AI interactions with the instruction-following variants of Granite, we develop a structured interface that enhances the model's ability to follow directives as including follow ups. The structure is split in multiple sections, or turns, where knowledge is commonly formatted and then aggregated for the model. For basic uses, turns include sections for just the human and the model; however, for enterprise uses, flexibility was added so that turns can include system information, external tools and functions available, further context, and even other agents and their responses (see Table 3).

Additionally, we carefully study the impact of specific/control tokens used in the structure of the prompt template, as shown in Table 4. Through ablations studies of different tokenizers, we find most BPE tokenizers fail to create guaranteed boundary lines when simple separators such as a newline are used. This not only makes masking during supervised finetuning unreliable but also leaks prompt control sequences into the prompt's text resulting in unwanted behavior and subpar performance. As such, we add special tokens into the vocabulary that allow us to create a structured language in the prompt that guarantees a separation between the formatting of the prompt and the prompt text itself.

Specifically, during post-training, the model has learned multiple roles, including: **user** role for any human input or, the query in the instruction-following system, **assistant** which indicates any output

<|start_of_role|>user<|end_of_role|>**What is 1+1?**<|end_of_text|>
<|start_of_role|>assistant<|end_of_role|>**2**<|end_of_text|>
<|start_of_role|>system<|end_of_role|>**Your name is Granite.**<|end_of_text|>
<|start_of_role|>available_tools<|end_of_role|>**[{"name": "get_temp", ...}, ...]**<|end_of_text|>
<|start_of_role|>user<|end_of_role|>**What is temperature in Boston?**<|end_of_text|>
<|start_of_role|>assistant<|end_of_role|><|tool_call|>**[{"name": "get_temp", ...}]**<|end_of_text|>
<|start_of_role|>tool_response<|end_of_role|>**{"temp": 20.5, "unit": "C"}**<|end_of_text|>

Table 3: Chat template for conversational tasks. Top of the table refers basic single turn chat between user and assistant, while the bottom shows an example of our model in tool use.

<|start_of_role|>	Denotes the start of a new turn, and precedes a role label.
<|end_of_role|>	Follows the role label and indicates the end of the turn's header.
<|end_of_text|>	Denotes the end of the turn and is trained as the end of sequence token.
<|tool_call|>	Produced by the model and is a designation that a function call follows

Table 4: Control tokens added to the vocabulary for user and assistant interactions.

generated by the model, **available_tools** turn for a structured list of tools and functions available to model to use, **tool_response** role of any external system that produces information based on called tools, and **system** role for guiding information that does not necessarily pertain to the user's query.

5.2 Supervised Finetuning

To enable instruction-following capabilities in our model, we further trained the pre-trained models using supervised finetuning with a curriculum-based approach where, in the first stage, we use all selected data and in the second phase, we focus on high quality, multiturn reasoning data with some data replay from the first stage. Furthermore, to support the model's ability to stop after a single turn in a multiple conversation, we unlearn the end of sequence token from pretrained model by setting its embedding weight to the arithmetic mean of the full embeddings vector. We then use the end of sequence token as the indicator for the end of a turn in the prompt structure.

We compute the loss at each step only on tokens produced by the model at any point in the prompt structure. That is, for multiturn samples, the loss is calculated for each model turn starting after the turn's header and until the next end of sequence token is reached. Everything else, including prompt control sequences, user and tools inputs, is masked out. To optimize training, we use variable length flash attention as well as the fit-first-decreasing bin packing algorithm to pack samples to create a near static batch size based on tokens instead of samples.

5.2.1 Data Mixture

Data mixture plays a critical role in supervised finetuning for improving model usability and general performance. We first create an internal set of benchmarks that focus on various domains useful for enterprise deployments of AI, including math, reasoning, and instruction following. We then create a static set of training examples as a baseline and mixed in one epoch of a single dataset to see how the model improves over that said baseline for each dataset in our finetuning mixture.

We filtered datasets that did not see improvement and grouped together datasets that showed improvement in the same domains. We then trained models with various sampling proportions to find a mixture of datasets with best average improvement over the baseline across the targeted domains. To find the optimal data mixture, we train for 5 million samples and test sampling rates that correspond to the range from a half epoch to four epochs of a dataset group. Note that we use a combination of only permissively licensed data from publicly available sources and internally collected SFT data, where each sample is formatted in form of instruction and response pairs (with optional context). Table 5 lists the overall statistics of our SFT data across six broad categories from general English to safety, used in Granite 3.0 post-training. See Section B.2 for the full list of individual datasets.

Domains	Sampling rate	$\overline{\#}$ turns	$\overline{\#}$ tokens/sample	Input/output tokens
General English	68%	1.33	670.6	0.61
Code	13%	1.15	647.0	0.65
Tools	8%	2.13	257.2	1.21
Math	6%	1.00	309.5	0.58
Multilingual	4%	1.75	1,722.3	0.33
Safety	1%	1.00	134.9	1.42

Table 5: Statistics of supervised finetuning data. Overall, our SFT data is largely comprised of three key sources: (1) publicly available datasets with permissive license, (2) internal synthetic data targeting specific capabilities, and (3) very small amounts of human-curated data.

5.2.2 Hyperparameter Search

We perform extensive search over the hyperparameters after the data mixtures are finalized. We search parameters such as learning rates, weight decay, warmup ratio, and batch size using the same metrics as the data mixture search. Ultimately, the first stage of SFT use a batch size of 1 million tokens with a learning rate in the magnitude of $1\mathrm{e}{-2}$. We employ a cosine decay schedule for the learning rate with a warmup ratio of 0.1 for 8b and 0.2 for 3b, and we decay to 0.1 of the peak learning rate for all models. Additionally, we set a constant weight decay of 0.1. For the second stage, we drop the batch size down to 32 thousand tokens, and the learning rate to magnitude of $1\mathrm{e}{-7}$. We train for 20 million samples in first stage and 100 thousand samples in stage two, which equates to approximately 30 thousand total training steps. We follow similar hyperparameters and settings for training both dense and MoE models.

5.3 Model Alignment

We further train our SFT models with reinforcement learning for human preference alignment. Specifically, the backbone of our model alignment is an unique combination of PPO (Rafailov et al., 2024; Korbak et al., 2022), BRAIn (Pandey et al., 2024), and Best-of-N Sampling, with an ensemble of reward models. Below we provide a brief description of the alignment data, followed by alignment techniques and different reward models with their ensemble used to align Granite 3.0 models.

5.3.1 Alignment Data

The alignment data composition plays a critical role in the usefulness and behavior of language models. Table 6 shows the composition of our data mixture that we used to align the Granite 3.0 models. We primarily use publicly available high quality datasets with permissible license including synthetic prompts tailored for improving specific capabilities like knowledge-based question and answering. We perform several small-scale experiments to find the optimal mixture across four key categories, such as general English, code, math and safety. This optimal mixture is used for all the algorithms described in the subsequent sections.

Table 6: Data mixture used for aligning the Granite SFT models.

Categories	Proportions	Datasets
General English	40%	HelpSteer2 [20], ShareGPT Prompts [21], Truthy-DPO [22], Synthetic Prompts
Code	25%	Synthetic Coding Prompts generated using modified OSS Instruct [23]
Math	5%	MetaMathQA [24]
Safety	30%	Anthropic-HH-RLHF [25]

[21] https://huggingface.co/datasets/nvidia/HelpSteer2
[22] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered
[23] https://huggingface.co/datasets/jondurbin/truthy-dpo-v0.1
[24] https://github.com/bigcode-project/starcoder2-self-align
[25] https://huggingface.co/datasets/meta-math/MetaMathQA

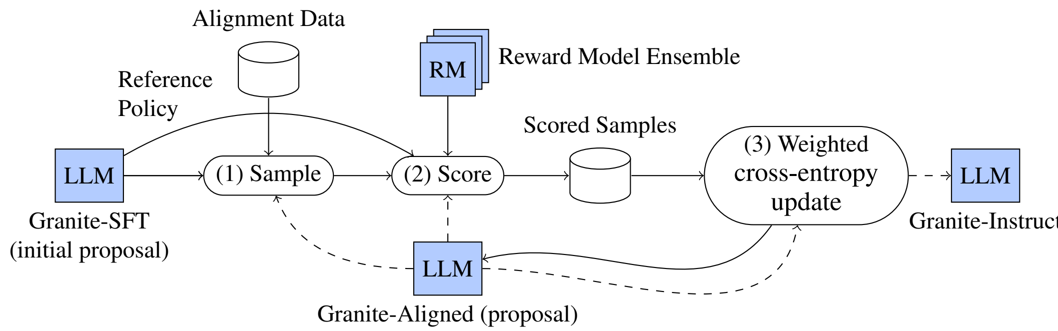


Figure 6: Granite 3.0 model alignment framework includes multiple iterations involving (1) Sampling, (2) Sample scoring optionally involving a proposal distribution, a reference distribution (SFT), an esemble of Reward Models and a Value Function. (3) SFT using a weighted cross-entropy loss. Process is repeated using updated model as new proposal (dashed line). The methods used in sucession are BRAIn, PPO and a last iteration of Best-of-N sampling.

5.3.2 Alignment Techniques

Following supervised finetuning, we employ model alignment techniques that rely on reward model(s) for supervision. The training objective aims at maximizing the expected reward, often augmented with a penalty term controlling the KL divergence of the learned policy from the initial SFT policy,

$$\mathbb{E}_{(x,y)\sim D_{\pi_\theta}}\left[r(x,y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{SFT}(y|x)}\right]$$

where r denotes the reward score, π_θ represents the policy being learned and π_{SFT} is the initial (instruct) model, serving as a baseline policy. β moderates the Kullback-Leibler divergence to prevent excessive deviation of π^θ from π^{SFT}. The optimal policy under this objective can be written as (Rafailov et al., 2024; Korbak et al., 2022):

$$\pi_\theta^* \propto \pi_{\text{SFT}}(y|x) \exp\left(\frac{r(x,y)}{\beta}\right) \tag{10}$$

Proximal Policy Optimization (PPO) learns a policy that minimizes the reverse KL-divergence to the above optimal policy. BRAIn, on the other hand, optimizes forward KL between optimal policy and current policy (Pandey et al., 2024) and results in very effective LLM alignment (Shen et al., 2024a). For aligning Granite 3.0 models, we combine the optimization of forward KL (BRAIn) and reverse KL (PPO) in a sequential manner – starting with a single iteration of Best-of-N (BoN) training, we first apply PPO followed by a short run of BRAIn training. We find this recipe to be quite effective for model alignment, achieving higher performance compared to any of the individual methods involved. See figure 6 for an illustration of our alignment framework.

Best-of-N Sampling. We use a single iteration of Best-of-N (BoN) sampling, as the first step in our model alignment pipeline. Training on BoN samples is by far the simplest yet very effective alignment technique (Stiennon et al., 2020; Sessa et al., 2024). We generate 64 responses for each sample and then rank them using an ensemble of the first two reward models from section 5.3.3. We experiment with both arithmetic mean and geometric mean to ensemble reward scores. Both methods perform similarly, with arithmetic mean giving more consistent gains across experiments.

Proximal Policy Optimization. PPO is a policy gradient method that employs a surrogate loss to efficiently minimize the reverse KL divergence to the optimal policy in 10. We use the trlX library (Havrilla et al., 2023) for PPO training; we modified the trlX to allow for LoRA training, with alpha and rank both set to 8. Each PPO run performs 1000 updates with a batch size of 8 and learning rate of 5e-7. The KL penalty coefficient is initialized at 0.05 and goes to a target value of 2 during the course of training. The number of rollouts is set to 64, with the reward as an arithmetic mean of normalized scores from three different reward models, described in section 5.3.3.

BRAIn. While PPO optimizes the reverse KL-divergence between the model and the target policy, BRAIn and its variants (Pandey et al., 2024; Wang et al., 2024b) optimize a self-normalized version of the forward-KL divergence and has been shown to achieve improved performance on several tasks such as abstractive summarization, helpfulness and chat benchmarks. The proposal distribution in BRAIn is updated after every 100 steps of training and the samples from the proposal distribution are labeled using an arithmetic mean of the reward models. These samples are then used for the next 100 steps of training. The cycle repeats until all the data available is exhausted.

5.3.3 Reward Models

Unlike most alignment approaches, in this work, we do not collect any human annotations over the model responses – to make up for the lack of direct human annotations, we instead resort to an ensemble of three vastly different rewarding mechanisms – 1) an aspect-level reward model akin to the SteerLM (Wang et al., 2024b), 2) a standard Bradley Terry reward model trained on preference pairs and 3) ratios of log-probabilities from two related models, as a contrastive reward signal. In the following, we discuss each of these in some detail followed by a discussion on ensembling strategies.

Multi-Aspect Reward Model. We train a multi-aspect regression-based reward model using Mistral-Nemo-Instruct [26] following the SteerLM recipe from HelpSteer2 (Wang et al., 2024b), where each model predicts the scalar value of the response's rating (a float ranging from 0 to 4) for each fine-grained aspect: Helpfulness, Correctness, Coherence, Complexity, and Verbosity. When using this reward model for alignment, the individual scores are collapsed into one score using the weights prescribed in (Wang et al., 2024b). These collapsed weight give a RewardBench score of 87.

Bradley-Terry Reward Model. We train an autoregressive reward model with the standard Bradley-Terry objective (Bradley & Terry, 1952; Rafailov et al., 2024) on pairs of preference data. The training data comprises one million preference pairs, including open-source gold preference data from various domains as well as synthetically generated preference pairs. For synthetic data generation, we adopt the model-gap strategy from (Naseem et al., 2024), where the accepted response comes from a strong model and the rejected response comes from a weaker model. We train a Mistral-7B-Instruct-v0.2 [27] on the whole preference data. The training hyper-parameters and the proportions of each data in the training mix are discussed in Appendix B.3.2. Our trained reward model gives a score of 84.5 on RewardBench.

Contrastive Reward Model. Two independent lines of prior work have shown contrastive log probabilities as an informative signal: First, in decoding research, a number of papers have shown that the difference between probabilities of a strong model and a related weak model can pick the next token more accurately than any of the two models being contrasted (Li et al., 2023e). Second, following the Direct Preference Optimization work (Rafailov et al., 2024), it has been shown that the sample level density ratio of the DPO instruct model with its corresponding base model gives high performance on RewardBench (Lambert et al., 2024). In this work, we contrast the Granite-3.0-8B-Instruct (SFT only) model with the Granite-3.0-2B-Instruct (SFT only) model and aggregate token level reward to get a sample level reward that is then used in the alignment algorithms.

Ensemble of Reward Models. When ensembling the reward models, we experiment with two simple approaches. 1) we compute the arithmetic mean of normalized reward scores, where each reward model's score is individually normalized using its mean and standard deviation over a range of samples. This approach can be used with all three alignment techniques. 2) we rank multiple responses for the same input separately using each reward signal, we then compute the score of each sample as the geometric mean of its ranks across reward models, the lower is better in this case. This approach is suited for best-of-N sampling and BRAIn.

[26] https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407

[27] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

5.4 Model Merging

We systematically train multiple models at each stage of the post-training pipeline, each specialized in a specific domain, such as multilingual understanding or reasoning. Before moving to the next stage of training and alignment, we merge the different model weights to create the best overall model across the tasks. The model's performance is validated on the same set of internal training benchmarks used in the data mixture and hyperparameter searches.

6 Infrastructure, Energy Consumption and Carbon Emissions

Infrastructure. We train the Granite 3.0 models using Blue Vela, one of IBM's supercomputing clusters building with NVIDIA H100 SuperPod and IBM Spectrum LSF. Each node in Blue Vela consists of dual 48-core processors, 8x NVIDIA H100 SXM5 80GB and 10 NVIDIA ConnectX-7 NDR InfiniBand Host Channel Adapters (HCA). Blue Vela employs 3.2Tbps InfiniBand interconnect to facilitate seamless communication between nodes, known for their high throughput and low latency. In addition, Blue Vela employs a separate, dedicated storage subsystem which is designed around the IBM Spectrum Scale ecosystem and the new IBM Storage Scale System 6000 (SSS) (Gershon et al., 2024). Utilizing InfiniBand and PCIe Gen 5 technology for optimal performance, each SSS appliance is capable of delivering upwards of 310 GB/s throughput for reads and 155 GB/s for writes. The Blue Vela cluster runs on 100% renewable energy to minimize the environmental impact.

Table 7: Energy consumption and carbon emissions of training Granite 3.0 models in the same data center. We take PUE of 1.3 and 0.39kg CO_2eq/KWh as carbon intensity factor.

Model	GPU power consumption	GPU-hours	Total power consumption (MWh)	Carbon (tCO_2eq)
Granite 3.0 2B	700W	192,030	174.6	68.1
Granite 3.0 8B	700W	832,102	757.0	295.2
Granite 3.0 1B-400M	700W	71,171	64.5	25.2
Granite 3.0 3B-800M	700W	133,308	121.2	47.2

Energy Consumption and Carbon Emissions. All our Granite 3.0 models are trained using a compute budget of 8.35×10^{23} FLOPS. This training has consumed energy resulting in emission of carbon dioxide. We show the energy consumption and carbon emission in Table 7. To calculate Watt-hour, we follow Touvron et al. (2023) which uses the formula:

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

where Power Usage Effectiveness (PUE) is set with 1.3. To calculate the emission we use the US national average carbon intensity factor of 0.39 kg CO_2eq/KWh according to U.S. Energy Information Administration[28] without taking location of data centers in consideration. A number of mitigation strategies can be used to reduce the energy and carbon footprint of training future Granite models. For example, the amount of resources used in training may be adjusted as a function of the availability of renewable energy, or the resources usage may be capped to not exceed certain energy usage or emissions limits. Moreover, we hope that releasing all our Granite 3.0 models in open source will help to reduce future carbon emission since the training is already done, and the models are relatively small and can be run on a single GPU (maximum 8B params).

7 Evaluation

We compare our Granite 3.0 models (base and instruct) against the open source models with similar active parameters from the following releases: Llama 3.1 and 3.2 (Dubey et al., 2024), Gemma-2 (Team et al., 2024), Mistral (Jiang et al., 2023), SmolLM [29]. All benchmark scores for baseline

[28] https://www.eia.gov/tools/faqs/faq.php?id=74&t=11
[29] https://huggingface.co/blog/smollm

models were also evaluated by ourselves using LM evaluation harness (Gao et al., 2024) using the same pipeline and configuration to avoid discrepancy between scores. Unless otherwise noted, all the results presented in this section apply to instruct models (except Table 8 and Table 9), and we often refer Granite-3.0-Instruct as Granite-3.0 for simplicity.

7.1 Pre-trained Language Model

Benchmark	**Metric**	**Gemma-2**	**Llama-3.2**	**Granite-3.0**	**Mistral**	**Llama-3.1**	**Granite-3.0**
Parameters		2B	3B	2B	7B	8B	8B
Human Exams							
MMLU	5-shot	53.01	**56.16**	55.00	62.33	**65.95**	65.54
MMLU-Pro	5-shot	21.97	**24.98**	23.79	29.49	32.60	**33.27**
AGI-Eval	5-shot	21.47	**24.40**	22.56	25.34	33.44	**34.45**
Commonsense							
WinoGrande	5-shot	71.59	71.59	**74.90**	78.37	79.24	**80.90**
OBQA	0-shot	41.80	**43.00**	**43.00**	44.20	44.40	**46.80**
SIQA	0-shot	52.66	39.69	**59.84**	39.38	53.90	**67.80**
PIQA	0-shot	79.11	77.48	**79.27**	82.15	81.18	**82.32**
Hellaswag	10-shot	74.66	76.39	**77.65**	83.01	81.70	**83.61**
TruthfulQA	0-shot, mc2	36.27	39.21	**39.90**	42.58	45.25	**52.89**
Reading Comprehension							
BoolQ	5-shot	78.59	74.28	**81.35**	84.28	85.63	**86.97**
SQuAD 2.0	0-shot	18.36	17.84	**25.22**	20.96	24.09	**32.92**
Reasoning							
ARC-C	25-shot	53.33	50.34	**54.27**	60.15	57.68	**63.40**
GPQA	0-shot	24.66	28.86	**30.58**	29.61	28.78	**32.13**
BBH	3-shot	36.45	39.54	**40.69**	44.99	46.42	**49.31**
MUSR	0-shot	**41.27**	35.58	34.34	40.74	37.96	**41.08**
Code							
HumanEval	pass@1	18.90	17.68	**38.41**	27.44	31.71	**52.44**
MBPP	pass@1	27.40	33.40	**35.40**	37.40	37.60	**41.40**
Math							
GSM8K	5-shot	23.88	25.17	**47.23**	36.85	50.64	**64.06**
MATH	4-shot	14.88	6.86	**19.46**	12.60	22.90	**29.28**
Average							
All		41.59	41.18	**46.47**	46.41	49.53	**54.77**
Open Leaderboard							
Open LLM Leaderbroad 1		52.12	53.14	**58.16**	60.55	63.41	**68.40**
Open LLM Leaderbroad 2		24.56	23.96	**28.80**	28.29	27.46	**34.89**

Table 8: Base version performance for Granite-3.0 dense and baseline models. The Open LLM Leaderboard 1 and 2 results are the average of tasks and metrics specified by the respective leaderboard.

To compare our base models with the current state-of-the-art, we evaluate Granite 3.0 on a large number of standard benchmark evaluations shown in Table 8 and Table 9. These evaluations cover six top-level categories: humane exams (MMLU (Hendrycks et al., 2020b), MMLU-Pro (Wang et al., 2024a), AGI-Eval (Zhong et al., 2024) (English only)), commonsense (WinoGrande (Sakaguchi et al., 2021), OBQA (Mihaylov et al., 2018), SIQA (Sap et al., 2019), PIQA (Bisk et al., 2020), Hellaswag (Zellers et al., 2019), TruthfulQA (Lin et al., 2022)), reading comprehension (BoolQ (Clark et al., 2019), SQuAD 2.0 (Rajpurkar et al., 2018)), reasoning (ARC-C (Clark et al., 2018), GPQA (Rein et al., 2023), BBH (Suzgun et al., 2022), MUSR (Sprague et al.)), code (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021)), and math (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021)), including two Hugging Face's Open LLM leaderboards.

Experiment results show that our Granite base models outperform baseline models in the vast majority of benchmarks. Looking at the average performance, the Granite-3.0-8B model outperforms Llama-3.1 8B and Mistral 7B by a convincing margin. More interestingly, the Granite-3.0-2B dense model achieves comparable performance as Mistral 7B, and the Granite-3.0-A800M-3B model outperforms

Benchmark	Metric	SmolLM	Granite-3.0	Llama-3.2	SmolLM	Granite-3.0
Active parameters		360M	400M	1B	1.7B	800M
Total parameters		360M	1B	1B	1.7B	3B
Human Exams						
MMLU	5-shot	**26.01**	25.69	30.79	28.80	**48.64**
MMLU-Pro	5-shot	11.24	**11.38**	11.78	11.55	**18.84**
AGI-Eval	5-shot	19.21	**19.96**	19.68	19.06	**23.81**
Commonsense						
WinoGrande	5-shot	57.22	**62.43**	60.69	60.93	**65.67**
OBQA	0-shot	37.60	**39.00**	37.20	42.00	**42.20**
SIQA	0-shot	34.88	**35.76**	34.42	34.63	**47.39**
PIQA	0-shot	71.33	**75.35**	74.59	76.06	**78.29**
Hellaswag	10-shot	53.45	**64.92**	63.66	65.74	**72.79**
TruthfulQA	0-shot,mc2	38.02	**39.49**	37.67	38.50	**41.34**
Reading Comprehension						
BoolQ	5-shot	62.69	**65.44**	66.12	68.96	**75.75**
SQuAD 2.0	0-shot	3.15	**17.78**	10.17	11.47	**20.96**
Reasoning						
ARC-C	25-shot	35.92	**38.14**	36.26	46.42	**46.84**
GPQA	0-shot	**26.26**	24.41	23.57	22.82	**24.83**
BBH	3-shot	26.20	**29.84**	30.76	29.30	**38.93**
MUSR	0-shot	**41.01**	33.99	34.39	33.99	**35.05**
Code						
HumanEval	pass@1	10.98	**21.95**	16.46	21.34	**26.83**
MBPP	pass@1	13.80	**23.20**	22.20	29.20	**34.60**
Math						
GSM8K	5-shot	1.36	**19.26**	6.90	6.60	**35.86**
MATH	4-shot	1.08	**8.96**	1.82	3.18	**17.40**
Average						
All		30.25	**34.57**	32.83	34.40	**42.05**
Open Leaderboard						
Open LLM Leaderbroad 1		35.84	**42.11**	40.47	42.16	**53.26**
Open LLM Leaderbroad 2		**22.08**	21.49	19.30	20.60	**25.01**

Table 9: Base version performance for Granite-3.0 MoE and baseline models

Gemma-2 2B and Llama-3.2 3B. Both models use much less computation than the respective baseline models. These results suggest a promising future in which we can not only get stronger AI models by scaling up computing but also by developing new techniques at all fronts, including architecture, data, and optimization. That being said, the Llama-3.2 3B models outperform the Granite-3.0-2B model on all Human Exams tasks, suggesting that around 40% more parameters and compute could still have an important impact on model performance. Furthermore, we observe that Granite models have a stronger lead in code and math domains. Considering the architectural similarity, the main difference in these available models is in the training data. This suggests that our data mixture is well optimized for improvements on a variety of different domains.

7.2 Post-trained Language Model

We compare our Granite 3.0 post-trained models (both dense and MoEs) on benchmarks across different capabilities, such as general knowledge and instruction following, function calling, RAG, and cybersecurity, including extensive safety evaluations.

Benchmark	Metric	Gemma-2	Llama-3.2	Granite-3.0	Mistral	Llama-3.1	Granite-3.0
Parameters		2B	3B	2B	7B	8B	8B
Instruction Following							
IFEval	0-shot	**53.83**	51.00	46.07	49.93	50.37	**52.27**
MT-Bench		7.91	**8.04**	7.66	7.62	8.21	**8.22**
Human Exams							
AGI-Eval	5-shot	**30.94**	30.82	29.75	37.15	**41.07**	40.52
MMLU	5-shot	56.83	**59.68**	56.03	62.01	**68.27**	65.82
MMLU-Pro	5-shot	27.19	**30.06**	27.92	30.34	**37.97**	34.45
Commonsense							
OBQA	0-shot	**44.20**	36.00	43.20	**47.40**	43.00	46.60
SIQA	0-shot	60.83	57.98	**66.36**	59.64	65.01	**71.21**
Hellaswag	10-shot	71.21	73.47	**76.79**	**84.61**	80.12	82.61
WinoGrande	5-shot	68.90	70.17	**71.90**	**78.85**	78.37	77.51
TruthfulQA	0-shot	53.17	49.71	**53.37**	59.68	54.07	**60.32**
Reading Comprehension							
BoolQ	5-shot	84.37	80.46	**84.89**	87.34	87.25	**88.65**
SQuAD 2.0	0-shot	16.21	**22.39**	19.73	18.66	21.49	**21.58**
Reasoning							
ARC-C	25-shot	**57.42**	30.47	54.35	63.65	60.67	**64.16**
GPQA	0-shot	**29.36**	29.19	28.61	30.45	32.13	**33.81**
BBH	3-shot	43.48	**43.92**	43.74	46.73	50.81	**51.55**
Code							
HumanEvalSynthesis	pass@1	40.55	45.12	**50.61**	34.76	63.41	**64.63**
HumanEvalExplain	pass@1	14.33	19.66	**45.58**	21.65	45.88	**57.16**
HumanEvalFix	pass@1	44.21	**55.18**	51.83	53.05	**68.90**	65.85
MBPP	pass@1	34.00	40.20	**41.00**	38.60	**52.20**	49.60
Math							
GSM8k	5-shot, cot	30.86	58.45	**59.66**	37.68	65.04	**68.99**
MATH	4-shot	21.76	**31.36**	23.66	13.10	**34.46**	30.94
Multilingual							
PAWS-X (7 langs)	0-shot	57.02	53.19	**61.42**	56.57	64.68	**64.94**
MGSM (6 langs)	5-shot	28.53	20.73	**37.13**	35.27	43.00	**48.20**
Average							
All		42.48	43.36	**47.01**	45.86	52.87	**54.33**
Open Leaderborads							
Open LLM Leaderbroad 1		58.70	61.47	**62.26**	65.54	68.58	**69.04**
Open LLM Leaderbroad 2		33.70	**33.73**	31.38	34.61	37.28	**37.56**
LiveBench		20.70	**22.90**	19.30	22.40	**27.60**	26.20
MixEval		**66.20**	65.20	64.80	73.55	73.35	**76.55**

Table 10: Instruct version performance of Granite-3.0 dense and baseline models.

7.2.1 General Knowledge and Instruction Following

Similar to pre-training, we show results of post-trained models on a broad range of standard benchmarks covering the earlier six categories from pre-training, including two additional categories such as instruction following (IFEval (Zhou et al., 2023), MT-Bench (Zheng et al., 2023)), and multilingual (PAWS-X (Yang et al., 2019), MGSM (Shi et al., 2022)) with a more comprehensive code evaluation (HumanEvalExplain (Python) and HumanEvalFix (Python) (Muennighoff et al., 2023a)). Note that HumanEvalSynthesize refers to standard HumanEval score, which measures Python code generation abilities. For PAWS-X, we report average results across English, German, French, Spanish, Japanese, Korean, Chinese, while in MGSM, we average across English, German, French, Spanish, Japanese, Korean, Chinese. For MT-Bench, we report average results of 5 runs using GPT-4 as a judge.

The general performance of instruction models is shown in Table 10 and 11. Experiment results show that Granite-3.0 models still consistently outperform baseline models on most tasks. The Granite-3.0-8B model achieves strong performance across different domains, making it a versatile tool for different enterprise use cases. Despite being a smaller model, the Granite-3.0-2B model outperforms Mistral-7B models on Code, Math, and Multilingual tasks, making it a good and economical choice

Benchmark	Metric	SmolLM	Granite-3.0	Llama-3.2	SmolLM	Granite-3.0
Active parameters		360M	400M	1B	1.7B	800M
Total parameters		360M	1B	1B	1.7B	3B
Instruction Following						
IFEval	0-shot	14.98	**32.39**	41.68	9.20	**42.49**
MT-Bench		3.49	**6.17**	5.78	4.82	**7.02**
Human Exams						
AGI-Eval	5-shot	18.25	**20.35**	19.63	19.50	**25.70**
MMLU	5-shot	26.05	**32.00**	45.40	28.47	**50.16**
MMLU-Pro	5-shot	11.05	**12.21**	19.52	11.13	**20.51**
Commonsense						
OBQA	0-shot	37.20	**38.40**	34.60	39.40	**40.80**
SIQA	0-shot	33.23	**47.55**	35.50	34.26	**59.95**
Hellaswag	10-shot	51.94	**65.59**	59.74	62.61	**71.86**
WinoGrande	5-shot	55.96	**61.17**	61.01	58.17	**67.01**
TruthfulQA	0-shot	39.98	**49.11**	43.83	39.73	**48.00**
Reading Comprehension						
BoolQ	5-shot	63.18	**70.12**	66.73	69.97	**78.65**
SQuAD 2.0	0-shot	**7.63**	1.27	16.50	**19.80**	6.71
Reasoning						
ARC-C	25-shot	38.48	**41.21**	41.38	45.56	**50.94**
GPQA	0-shot	**25.34**	23.07	25.67	25.42	**26.85**
BBH	3-shot	30.48	**31.77**	33.54	30.69	**37.70**
Code						
HumanEvalSynthesis	pass@1	11.59	**30.18**	35.98	18.90	**39.63**
HumanEvalExplain	pass@1	7.77	**26.22**	21.49	6.25	**40.85**
HumanEvalFix	pass@1	4.27	**21.95**	**36.62**	3.05	35.98
MBPP		14.40	**15.40**	**37.00**	25.20	27.40
Math						
GSM8k	5-shot,cot	0.30	**26.31**	26.16	0.61	**47.54**
MATH	4-shot	0.18	**10.88**	17.62	0.14	**19.86**
Multilingual						
PAWS-X (7 langs)	0-shot	5.03	**45.84**	34.44	17.86	**50.23**
MGSM (6 langs)	5-shot	0.13	**11.80**	23.80	0.07	**28.87**
Average						
All		21.34	**31.35**	34.07	24.82	**40.20**
Open Leaderboards						
Open LLM Leaderbroad 1		35.58	**46.76**	47.36	39.87	**55.83**
Open LLM Leaderbroad 2		19.85	**23.46**	26.50	18.30	**27.79**
LiveBench		3.40	**10.40**	11.60	3.40	**16.80**

Table 11: MoE Instruction Models

for many domain-specific use cases. Furthermore, the Granite MoE models outperform the baseline models with a significant margin on the majority of metrics. The Granite-3.0-A800M-3B achieved comparable performance as Gemma-2-2B and Llama-3.2-3B. The Granite-3.0-A400M-1B achieved comparable performance as Llama-3.2-1B and outperformed SmolLM-1.7B. The strong performance, combined with very low computing requirements (400M and 800M), makes them very attractive options for edge devices, including mobile phones and smart watches. However, the diminished gap also suggests that we could further improve our instruction tuning ability. The instruction tuning for MoE also remains an open challenge.

Benchmark	Gemma-2 2B	Llama-3.2 3B	Granite-3.0 2B	Mistral 7B	Llama-3.1 8B	Granite-3.0 8B
BFCL V2	29.85	**65.82**	64.00	46.70	67.02	**69.19**
ToolAlpaca	17.00	38.00	**42.00**	34.00	37.00	**39.00**
Nexus	20.40	**50.60**	48.70	59.70	**64.20**	60.70
API Bank	21.80	45.90	**66.90**	60.90	**68.20**	63.40
SealTools	7.70	**41.80**	36.40	48.00	37.30	**51.40**
API Bench	12.75	6.11	**13.91**	12.40	16.32	**25.46**
Average	18.25	41.37	**45.32**	43.61	48.34	**51.52**

Table 12: Performance comparison of Granite 3.0 dense models with models of comparable sizes on the function calling benchmarks. Granite models outperform other models in their category by a significant margin.

Benchmark Active parameters Total parameters	SmolLM 360M 360M	GraniteMoE 400M 1B	Llama-3.2 1B 1B	SmolLM 1.7B 1.7B	GraniteMoE 800M 3B
BFCL V2	10.00	43.83	21.44	10.00	41.11
ToolAlpaca	–	22.00	1.00	–	37.00
Nexus	–	29.20	5.30	–	28.90
API Bank	–	41.10	9.30	–	63.70
SealTools	–	15.00	1.90	–	24.20
API Bench	1.41	12.37	4.13	3.06	6.86

Table 13: Performance comparison of Granite 3.0 MoE models with models of comparable sizes on function calling benchmarks. Models with unavailable results have context lengths that are too small for this evaluation. Granite MoEs consistently outperform SmolLM and LLama-3.2-1B on all the six benchmarks.

7.2.2 Function (Tool) Calling

Function calling tasks evaluate the LLM's ability to effectively use external APIs/tools to perform user-specified tasks. We evaluate function calling capabilities of different models using the following public benchmarks, namely Berkeley Function-Calling Leaderboard[30] (BFCL-V2), API-Bank (Li et al., 2023c), API-Bench (Patil et al., 2023), ToolAlpaca (Tang et al., 2023), Nexus (Srinivasan et al., 2023), SealTools (Wu et al., 2024), and API-Bench (Patil et al., 2023).

BFCL-V2 contains 3,951 tool-calling test examples divided into the following two categories (a) Python: Simple Function, Multiple Function, Parallel Function, and Parallel Multiple Function, and (b) Non-python: Chatting Capability, Relevance/Irrelevance Detection, REST API, SQL, Java, and Javascript. **API-Bank** has 314 dialogues with 753 API calls to evaluate LLMs' capabilities in planning, retrieving, and calling APIs. We report numbers for API-Bank level-1 which has a total of 399 test samples that test abilities to find the right set of APIs and its parameters from a specified list of possible APIs. **ToolAlpaca** is a synthetic data generation approach containing 271 tool-use instances spanning 50 distinct categories. We use the simulated part of ToolAlpaca which has a total of 100 test examples. **Nexus**[31] is another function calling test set with a total of 318 test examples covering 65 different APIs. **SealTools** is a new synthetic dataset with a separate out-of-domain test set containing 654 diverse examples that use tools from a pool of size more than 4,000 tools. This test set also includes a small pool of 27 examples that require nested API calling (output of one API is used as input to the next). Since not all the models were trained with this capability and we are testing models in a zero-shot mode, we filter the 27 nested samples and report metrics on the remaining 627 instances that have single and multiple sequential tool calls. **API-Bench** tests the model's capability to generate a single line of code for Torchhub, Huggingface, and Tensorhub APIs. We evaluate the models by leveraging the API references obtained using BM25 retriever.

BFCL reports *Overall Accuracy* which is a weighted average of different metrics such as Abstract Syntax Tree (AST) summary, execution summary, relevance, and irrelevance detection. In API-Bench, we use *AST tree accuracy score* as the metric. For the rest of the datasets, we report an "Exact Match"

[30] https://gorilla.cs.berkeley.edu/blogs/12_bfcl_v2_live.html

[31] https://huggingface.co/datasets/Nexusflow/NexusRaven_API_evaluation

score which checks if the predicted APIs and their parameters are exact matches of the gold and in the right sequence. All evaluations for all models are done in a zero-shot manner.

Tables 12 and 13 show the function calling results. On average, both the 2B and 8B models outperform the other models we compare against, demonstrating their strong function calling capabilities often critical for building agentic systems. E.g., average +3.22% over Llama-3.1-8B-Instruct, indicating the effectiveness of our well-curated function calling data for improving specific capabilities in model training. Interestingly, our dense 2B model outperforms our 8B model on API Bank. Our small MoE models are also showing very strong performance, not only outperforming SmolLM models but also SOTA models like LLama-3.2-1B-Instruct across all the benchmarks. E.g., GraniteMoE 3B-A800M Instruct achieves average 33.62%, while Llama-3.2-1B-Instruct is only able to obtain 7.17% on 6 benchmarks (see Table 13). This shows that despite being small, our Granite MoE models are not only good for conversations but also capable of effectively calling functions and tools.

7.2.3 CYBERSECURITY

We evaluate our models on cybersecurity tasks using the evaluation benchmark constructed by Levi et al. (2024). This benchmark includes of an extensive set of cybersecurity tasks (internal) alongside other publicly available security benchmarks. Specifically, the IBM internal subset of this benchmark comprises of 8 tasks, namely Adversarial MITRE ATT&CK, SIEM Rule TTP Mapping, CTI Detection and Mitigation Mapping, CWE Technical Impact Mapping, CTI Relationship Prediction, CTI Entity Classification, MITRE ATTT&CK Entity Classification, and CWE Description Summarization. The public subset consists of 7 tasks in total that includes SecEval (Li et al., 2023a), CISSP Assessment Questions, Cybersecurity Skill Assessment, CyberMetric (Tihanyi et al., 2024), Cyber Threat Intelligence Multiple Choice Questions (CTI-MCQ) (Alam et al., 2024), Cyber Threat Intelligence Root Cause Mapping (CTI-RCM) (Alam et al., 2024), and MMLU Computer Security (SecMMLU) (Hendrycks et al., 2020a). See CyberPal.AI (Levi et al., 2024) for more details.

We evaluate the CWE Description Summarization task using ROUGE scores (ROUGE-1, ROUGE-2, and ROUGE-L) (Lin, 2004). For all other tasks, we measure performance using accuracy. The reported results, averaged over internal and public benchmarks, are presented in Tables 14, 15 for both dense and MoE models in the cybersecurity domain. All of our models consistently outperform their counterpart models in this benchmark, demonstrating their effectiveness in complex, domain-specific tasks, such as cybersecurity, which are of great importance in enterprise contexts.

Benchmark	**Gemma-2** 2B	**Llama-3.2** 3B	**Granite-3.0** 2B	**Mistral** 7B	**Llama-3.1** 8B	**Granite-3.0** 8B
Public (7 Tasks)	55.08	58.73	**64.31**	64.86	71.31	**71.89**
Internal (8 Tasks)	53.31	54.30	**66.78**	56.08	57.04	**68.88**
Overall (15 Tasks)	54.13	56.37	**65.63**	60.17	63.70	**70.28**

Table 14: Performance comparison of Granite dense models on the cybersecurity benchmark.

Benchmark	**SmolLM**	**GraniteMoE**	**Llama-3.2**	**SmolLM**	**GraniteMoE**
Active parameters	360M	400M	1B	1.7B	800M
Total parameters	360M	1B	1B	1.7B	3B
Public (7 Tasks)	20.95	**34.53**	46.25	24.55	**60.65**
Public (8 Tasks)	30.22	**41.26**	43.45	34.07	**58.61**
Overall (15 Tasks)	25.90	**38.12**	44.76	29.63	**59.56**

Table 15: Performance comparison of MoE models with models of comparable sizes on the Cyber-security benchmark. Granite models performs the best in their category, significantly outperforming other models.

7.2.4 RETRIEVAL AUGMENTED GENERATION

Given the question and the relevant document in the context, we evaluate our models to generate factually correct and relevant answers. To evaluate our model's RAG capabilities, we make use of the test sets of the RAGBench (Friel et al., 2024) dataset and the RAG assessment (RAGAS) evaluation

Dataset	Gemma-2 2B		Llama-3.2 3B		Granite-3.0 2B		Mistral 7B		Llama-3.1 8B		Granite-3.0 8B	
	F↑	C↑	F↑	C↑	F↑	C↑	F↑	C↑	F↑	C↑	F↑	C↑
CovidQA	75.76	**64.18**	77.96	62.34	**81.62**	63.49	83.61	63.49	82.58	62.12	**86.80**	**66.73**
DelucionQA	80.62	**63.35**	83.51	63.24	**84.51**	61.48	**88.32**	60.93	87.49	**69.44**	85.13	67.97
EManual	76.78	**66.33**	74.61	64.63	**85.08**	63.93	74.89	63.55	**87.40**	68.09	85.67	**68.24**
ExpertQA	53.66	**59.57**	**62.43**	58.60	58.08	58.87	65.69	61.28	62.15	59.89	**68.90**	**61.35**
HAGRID	**83.64**	63.42	82.10	63.32	82.84	**66.60**	**84.94**	66.92	84.38	62.07	81.95	**70.58**
HotpotQA	84.98	74.16	80.36	72.19	**88.55**	**76.26**	86.90	77.09	80.77	69.29	**89.48**	**77.85**
MS Marco	79.00	63.66	82.80	62.38	**85.83**	**65.46**	82.99	63.83	86.82	62.89	**90.23**	**68.48**
PubMedQA	72.64	63.68	70.32	64.58	**83.69**	**66.03**	80.19	66.83	73.34	64.48	**89.68**	**68.29**
TAT-QA	67.63	64.40	75.22	65.14	**76.12**	**70.85**	75.74	68.14	83.14	66.89	**85.82**	**76.38**
TechQA	32.34	**41.51**	**61.38**	41.11	34.35	40.26	**71.01**	43.46	58.07	**45.64**	33.85	43.45
FinQA	52.08	47.12	57.65	52.57	**63.00**	**56.26**	62.37	57.59	**72.34**	58.84	66.02	**58.96**
Average	69.01	61.03	73.48	60.92	**74.88**	**62.68**	77.88	63.01	78.04	62.69	**78.50**	**66.21**

Table 16: Performance of different models on RAGBench using RAGAS evaluation framework. Numbers shows average of 3 runs with GPT-4 as the judge. Following (Roychowdhury et al., 2024), we report faithfulness and correctness as they are better aligned with human expert judgment.

framework (Es et al., 2023) to evaluate our models. For the test sets in RAGBench, we use the outputs generated by the GPT-3.5 model. Table 16 shows the results on RagBench. We use GPT-4 as the LLM judge to evaluate the models. The metrics we use to evaluate the models are:

- Faithfulness: It measures the factual consistency of the generated answer wrt the given context and is computed from the answer and the retrieved context.
- Correctness: It measures answer correctness compared to the ground truth response as a combination of factuality and semantic similarity.

On average, our Granite dense models outperform their counterparts with similar parameter size (e.g., Granite-3.0-8B-Instruct achieves $+4\%$ in correctness Llama-3.1-8B-Instruct). Both our 8B and 2B models outperform other models on the MS MARCO task, a reading comprehension task, by a large margin. However, on the TechQA task, we underperform other models. This might suggest further collection of data from the technical support domain may be required.

7.2.5 Safety

Moodels	BOLD (↓)	CrowS-Pairs (↓)	ALERT (↑)	SALAD-Bench (↑)
Mistral-7B-Instruct	55.46	**62.31**	96.48	74.41
Llama-3.1-8B-Instruct	46.43	64.04	**99.05**	90.51
Granite-3.0-8B-Instruct	**46.14**	63.44	98.89	**95.30**

Table 17: Safety benchmark results of Granite-3.0-8B-Instruct with models of comparable sizes.

We evaluate our Granite 3.0 models on five academic benchmarks covering diverse aspects of safety to ensure that it avoids generating harmful, illegal, or unethical content, while still maximizing helpful information. The safety benchmarks are described as follows.

AttaQ (Kour et al., 2023) is a semi-automatically curated dataset, consisting of adversarial question attack samples representing queries that the LLMs must refrain from answering. It consists of 7 categories of harmful input prompts – Harmful Info, PII, Substance Abuse, Explicit Content, Violence, Discrimination, and Deception. To evaluate the model's responses to input prompts belonging to each of the above-mentioned categories, we make use of a reward model judge[32].

BOLD (Bias in Open-ended Language Generation Dataset) (Dhamala et al., 2021) is a dataset to evaluate fairness in open-ended language generation in English language. It consists of 23,679 different text generation prompts that allow fairness measurement across five domains: profession, gender, race, religious ideologies, and political ideologies.

[32] `https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2`

CrowS-Pairs (Nangia et al., 2020) is a challenging dataset for measuring the degree to which stereotypical biases present in the language models. It consists of 1508 examples that cover stereotypes dealing with nine types of bias, like race, religion, and age.

ALERT (Tedeschi et al., 2024) is a large-scale benchmark to test how safe LLMs are by evaluating them based on a novel fine-grained risk taxonomy (consisting of 6 macro and 32 micro categories). We evaluate the models on 14,763 test prompts containing a mix of different categories. The responses generated by the LLMs are determined as ”safe” or ”unsafe” using LlamaGuard-7B[33].

SALAD-Bench Li et al. (2024b) is a comprehensive benchmark designed for evaluating LLMs focusing on both attack and defense methods. It features a hierarchical structure with three levels, encompassing 6 domains, 16 tasks and 66 categories, allowing an in-depth assessment of safety. We test our model on the base set of SALAD-Bench, which consists of 21,318 questions aimed at assessing safety. Model-generated responses are evaluated by MD-Judge, a LLM judge proposed by SALAD-Bench, which classified answers as either ”safe” or ”unsafe”. We then calculate the safety rate, as the percentage of safe responses.

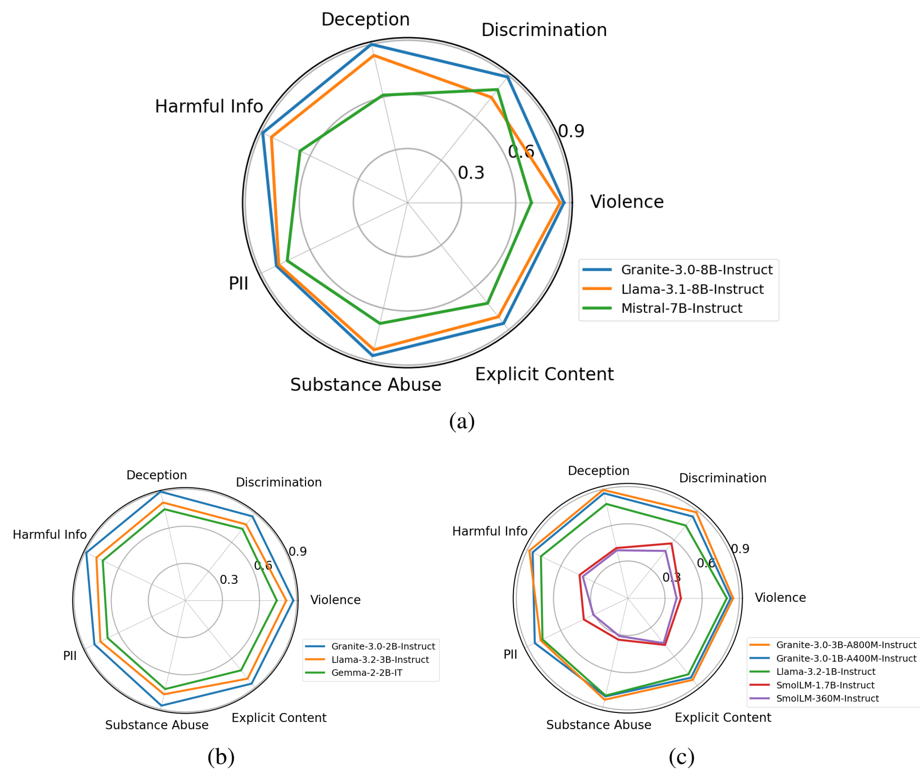


Figure 7: Comparison of the harmlessness scores of different models across various harm types on AttaQ benchmark. (a) 8B parameter models, (b) 2B-3B parameter models, (c) MoE models. Best viewed in color.

Figures 7(a), 7(b), 7(c) shows the radar plots of the different Granite 3.0 models for each of the AttaQ labels. Granite-3.0 models perform best in their parameter range, outperforming other models in all 7 aspects of safety, including Llama-3.1, Llama-3.2 and Gemma-2 models. Table 17 compares Granite-3.0-8B-Instruct, Llama-3.1-8B-Instruct, and Mistral-7B-Instruct on 4 additional safety benchmarks. Results show that our models achieve very competitive scores compared to Llama-3.1-8B-Instruct, demonstrating effectiveness of our safety alignment while still retaining helpfulness.

[33] https://huggingface.co/meta-llama/LlamaGuard-7b

8 Socio-Technical Harms and Risks

Numerous potential socio-technical harms and risks of LLMs have been identified in recent years, including misinformation, hallucination, lack of faithfulness or factuality, leakage of private information, plagiarism or inclusion of copyrighted content, hate speech, toxicity, human-computer interaction harms such as bullying and gaslighting, malicious uses, and adversarial attacks.

In line with the IBM AI Ethics Board, a cross-disciplinary body that defines the AI ethics vision and strategy for the IBM Corporation, we have followed several risk mitigation strategies while creating and releasing Granite 3.0 models. This includes comprehensive data governance which includes clearance, block-listing, filtering of documents with potential hate, abuse and profanity. Through model alignment with a dedicated focus on safety, we have also encouraged prosocial and less harmful model behavior with the aim to mitigate certain aspects of misuse and value alignment risks. We have also endeavoured to safeguard against some of the risks by assessing safety through standardized AI safety benchmarks which can be seen in Section 7, including internal red teaming to better understand the risks associated with external use of Granite 3.0 models in critical enterprise use cases. However, evaluating on benchmarks is only a limited approach for revealing socio-technical harms. Going forward, as the harms from LLMs become well-defined, or as Granite model capabilities advance, we will explore extended training and staged releases to further minimize risks.

In addition, as part of IBM's commitment to responsible AI, we are also introducing a new family of LLM-based Granite Guardian guardrail models [34], providing the most comprehensive set of risk and harm detection capabilities available in the community today. These models can be used to monitor and manage inputs and outputs to any LLM, whether open or proprietary. We have also released a Responsible Use Guide [35] to support developers to build AI responsibly with our Granite 3.0 models. This encompasses a series of assets to help developers design and implement responsible AI best practices and keep their users safe. However, every enterprise often has its own regulations to conform to, whether they come from laws, social norms, industry standards, market demands, or architectural requirements; we believe that users should be empowered to personalize our released Granite models according to their own values (within bounds) (Kirk et al., 2023).

9 Conclusion

We present Granite 3.0, an openly available family of lightweight generative language models that are highly versatile in their ability to accomplish a wide range of enterprise tasks. We release four sizes of models across two different architectures (dense and mixture-of-experts), and provide both base and instruct checkpoints. Aligned with IBM's commitment to transparent and responsible AI, we present descriptions of training data, pre-processing steps, data mixture, training details, energy consumption, and evaluation methodologies used throughout the model development lifecycle. Granite 3.0 language models demonstrate strong performance across a battery of academic benchmarks for language understanding, reasoning, coding, function calling, and safety. Our experience and results demonstrate that Granite 3.0 language models have a proven ability to better handle different enterprise tasks such as RAG, cybersecurity and function calling among others. We release all our Granite 3.0 language models under an Apache 2.0 license for both research and commercial use. We plan to continuously release updates to these models to improve their performance with safety in mind, e.g., improving multilinguality and coding, including long-context model variants.

[34] https://huggingface.co/collections/ibm-granite/granite-guardian-66db06b1202a56cf7b079562
[35] https://www.ibm.com/granite/docs/resources/responsible-use-guide.pdf

A Contributions and Acknowledgements

Model Training

Aditya Prasad, Adriana Meza Soria, Anurag Roy, Ashish Agrawal, Gaoyuan Zhang, Gaurav Pandey, Keshav Ramji, Matt Stallone, Mayank Mishra, Rameswar Panda, Ramon Fernandez Astudillo, Shawn Tan, Tahira Naseem, Yikang Shen⋆.

Synthetic Data

Adriana Meza Soria, Asim Munawar, Atin Sood, Bridget McGinn, Chris Johnson, Christoph Tillmann, Chulaka Gunasekara, Gakuto Kurata, Geert Janssen, George Safta, Graeme Blackwood, Guy Uziel, Harsha Kokel, Hiroshi Kanayama, Ibrahim Abdelaziz, Ian Molloy, Issei Yoshida, Jian-Ming Xu, Juergen Bross, Kavita Srinivas, Kinjal Basu, Kshitij Fadnis, Luca Buratti, Masayasu Muraoka, Matan Levi, Maxwell Crouse, Mayank Agarwal, Osher Elhadad, Pavan Kapanipathi⋆, Sadhana Kumaravel, Sarath Swaminathan, Saurabh Pujar, Shirin Shorabi, Shyam Ramji, Siva Sankalp Patel, Subhajit Chaudhury, Tim Bula, Wei Zhang, Yair Allouche, Yara Rizk, Young-suk Lee, Youngja Park, Yousef Elkurdi.

Data Engineering and management

Aanchal Goyal, Abdulhamid Adebeyo, Alexandru Soicescu, Alexei Karve, Alexey Roytman, Alice Lee, Aliza Heching, Andrew Coleman, Anna Lisa Gentile, Basel Shbita, Benjamin Ruby, Bing Zhang, Bishwaranjan Bhattacharjee, Boris Lublinsky, Chad DeLuca, Changchang Liu, Chirag Garg, Christoph Tillman, Codrut Stoicescu, Constantin Adam, Daniel (Chung-hao) Tan, David Wood, Dhiraj Joshi, Eelaaf Zahid, Eric Butler, Gakuto Kurata, Guangjie Ren, Hajar Emami Gohari, Heiko Ludwig, Hima Patel, Hiroshi Kanayama, Hiroya Matsubara, Issei Yoshida, Jay Gala, Jorge Sanz, Juan Cappi, Juergen Bross, Julian Rossi, Kun-Lung Wu, Luis Bathen, Manuela Fernandes, Mark Lewis, Maroun Touma, Masayasu Muraoka, Matthew White, Mohammad Nassar, Nathalie Baracaldo, Nicolas Mello, Nitin Ramchandani, Paramaswaran Selvam, Pengyuan Li, Petros Zerfos⋆, Praneet Adusumilli, Radha Ratnaparkhi, Rakesh Jain, Revital Eres, Ryan Gordon, Saptha Surendran, Shafiq Abedin, Shalisha Witherspoon, Shanmukha Guttula, Shiqiang Wang, Shivdeep Singh, Shubhi Asthana, Susanna Munoz, Syed Zawad, Taiga Nakamura, Takuma Udagawa, Tom Griffin, Vaibhav Saxena, Xuan-Hong Dang, Yan Koyfman, Yang Zhao, Yi Zhou, Yousaf Shah, Yousef El-Kurdi, Yuan Chi Chang, Yuya Ong.

Evaluation and safety

Aditya Prasad, Ambrish Rawat, Ashish Agrawal, Ateret Anaby-Tavor, Derek Leist, Farhan Ahmed, George Kour, Hui Wu, Ioana Baldini, Ja Young Lee, Kate Soule, Kush Varshney⋆, Manish Nagireddy, Matan Levi, Matt Stallone, Moninder Singh, Nathalie Baracaldo Angel, Onkar Bhardwaj, Veronique Demers, Juergen Bross, Onkar Bhardwaj, Ateret Anaby-Tavor, Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Prasanna Sattigeri, Inkit Padhi, Inge Vejsbjerg.

Infrastructure

Andrew Figueroa, Aniruddha Narkhede, Bob Calio, Bob Krull, Brian Belgodere⋆, Carlos Fonseca, Colm Malone, Connor McStay, Drew Wyskida, Gary Leonardi, John Lewars, John Walter, Matthew Connolly, Michael Spriggs, Ray Rose, Robert Walkup.

Developer Documentation and Recipes

Adam Pingel⋆, Adriana Meza Soria, Amogh Ranavade, Anita Govindjee, Anna Gutowska, Anthony Annunziata, Anupam Chakraborty, Ash Minhas, BJ Hargrave, Connor Leech, Dave Nielsen, Dean Wampler, Fayvor Love, Gabe Goodhart, Helen Stanton, Hima Patel, Ibrahim Ibrahim, Joe Olson, Justin MacNair, Kelly McGowan, Lara Neich, Lauren McHugh, Lindsay Lehane, Mandana Vaziri, Martin Hickey, Raju Pavuluri, Rangeet Pan, Saurabh Sinhas, Shivdeep Singh, Shrinidhi Deshapande, Sujee Maniyam, Trevor Grant,

PROJECT MANAGEMENT

Abraham Daniels, Anita Govindjee, Derek Leist, Eda Kavlakoglu, Felix Eickhoff, Hui Wu, Kate Soule*, Lan Hoang, Luke Inglis.

GRANITE TECHNICAL LEADERSHIP

David Cox, Luis Lastras, Nirmit Desai, Rameswar Panda, Sandeep Gopisetty.

IBM RESEARCH LEADERSHIP

Dario Gil, Sriram Raghavan, Ruchir Puri, Aya Soffer, Mukesh Khare, Danny Barnett, Talia Gershon, Priya Nagpurkar, Nicholas Fuller, Jeff Welser, Susana Rodriguez de Tembleque.

ACKNOWLEDGMENTS

We would like to acknowledge the efforts of numerous teams at IBM Research AI and Hybrid Cloud Platform, IBM AI Infrastructure, IBM Software, IBM Legal, IBM Data and Model Governance, and IBM Brand, Marketing, and Communications teams. We would also like to thank Arvind Krishna for his steadfast support of this effort, as well as members of the IBM executive leadership team and corporate technology leadership team. Thanks and acknowledgement to Dakshi Agrawal, Bill Higgins, Carlos Costa, Mudhakar Srivatsa, Sahil Suneja, J.R. Rao, Darrell Reimer, Xuan Liu, Maja Vukovic, Ritika Gunnar, Armand Ruiz, Maryam Ashoori, Robert Stanich, Nisarg Patel, Vincent Nelis, Raghu Kiran Ganti, Betsy Greytok, John McBroom, Steven Purdy, Bryan Bortnick, Trent Gray-Donald, Laura Chiticariu, Terry Yoo, Amy Angelini, Nikhilesh Murthy, Brittany Forgione, and Steven Tomasco.

B DATA

We open-source our data curation recipes in Data Prep Kit [36] for all Web and other datasets. We point the reader to the implementations therein for specifics on our data curation recipes Wood et al. (2024)

B.1 PRE-TRAINING DATA

This section outlines the datasets employed during our both stages of pre-training. To recognize specific features, we utilize the following annotations:

- `P1`: dataset used in stage-1 pre-training.
- `P2`: dataset used in stage-2 pre-training.
- `ML12`: sources from which we selected data in the following languages: English, German, Spanish, French, Japanese, Portuguese, Arabic, Czech, Italian, Korean, Dutch, Chinese.
- `IBM-Curated`: IBM's curated compendium of unstructured language data and code files.
- `IBM-Synthetic`: synthetic data created by IBM.

B.1.1 WEB

- FineWeb [37] [P1]: The FineWeb dataset consists of more than 15T tokens of cleaned and deduplicated English web data from CommonCrawl (Penedo et al., 2024).
- Webhose [P1,P2][IBM-Curated]: Unstructured web content in English converted into machine-readable data feeds acquired by IBM.
- DCLM-Baseline[38] [P2]: This is a 4T token / 3B document pretraining dataset that achieves strong performance on language model benchmarks (Li et al., 2024a).

[36]https://github.com/IBM/data-prep-kit
[37]https://huggingface.co/datasets/HuggingFaceFW/fineweb
[38]https://huggingface.co/datasets/mlfoundations/dclm-baseline-1.0

B.1.2 Code

- Code Pile [P1,P2][IBM-Curated]: Our code pile is sourced from a combination of publicly available datasets like Github Code Clean[39], StarCoderdata[40], and additional public code repositories and issues from GitHub. We filter raw data to retain a list of 116 programming languages and only keep files with permissive licenses for model training.
- FineWeb-Code [P2][IBM-Curated]: FineWeb-Code contains programming/coding related documents filtered from the FineWeb dataset using a pipeline similar to FineWeb-Edu but using Mixtral-8x22B-Instruct for annotation (Penedo et al., 2024).
- CodeContests[41] [P2]: A competitive programming dataset for machine-learning. Problems include test cases in the form of paired inputs and outputs, as well as both correct and incorrect human solutions in a variety of languages (Li et al., 2022).

B.1.3 Domain

- USPTO [P1, P2][IBM-Curated]: Collection of US patents granted from 1975 to May 2023, excluding design patents.
- Free Law [P1, P2][IBM-Curated]: Public-domain legal opinions from US federal and state courts.
- Pubmed Central [P1, P2][IBM-Curated]: Biomedical and life sciences papers.
- EDGAR Filings [P1, P2][IBM-Curated]: Annual reports from all the publicly traded companies in the US spanning a period of more than 25 years.
- SEC Filings [P1, P2][IBM-Curated]: 10-K/Q filings from the US Securities and Exchange Commission (SEC) for the years 1934-2022.
- FDIC [P1, P2][IBM-Curated]: The data is from the annual submissions of the FDIC.
- Earnings Call Transcripts [P1, P2][IBM-Curated]: Transcripts from the quarterly earnings calls that companies hold with investors. The dataset reports a collection of earnings call transcripts, the related stock prices, and the sector index.
- IBM Documentation [P2][IBM-Curated]: IBM redbooks and product documents, as well as publicly available documentation from Fortune 500 companies.
- Cybersecurity [P2][IBM-Curated]: Compendium of data crawled from different Web sources about cybersecurity-related topics.

B.1.4 Multilingual

- Multilingual Wikipedia [P2][IBM-Curated, ML12]: Multilingual wikipedia data of 11 different languages that Granite models are trained.
- Multilingual Webhose [P2][IBM-Curated, ML12]: Unstructured multilingual web content converted into machine-readable data feeds acquired by IBM.
- MADLAD-12 [P2][ML12]: A document-level multilingual dataset filtered from MADLAD-400[42], covering 12 languages filtered out of 419 languages in total.

B.1.5 Instructions

- Code Instructions Alpaca[43] [P2]: A dataset of instruction-response pairs about code generation problems.
- Glaive Function Calling V2[44] [P2]: A function calling dataset in real-world scenarios.

[39] https://huggingface.co/datasets/codeparrot/github-code-clean
[40] https://huggingface.co/datasets/bigcode/starcoderdata
[41] https://huggingface.co/datasets/deepmind/code_contests
[42] https://huggingface.co/datasets/allenai/MADLAD-400
[43] https://huggingface.co/datasets/TokenBender/code_instructions_122k_alpaca_style
[44] https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2

- Self-OSS-Instruct-SC2[45]: A synthetic dataset curated from the Stack V1 pretraining dataset without any human annotations or distilled data from huge and proprietary LLMs.
- Glaive Code Assistant V3[46] [P2]: A dataset of approximately 1M code problems and solutions generated using Glaive's synthetic data generation platform. This source includes the first and second versions of the dataset.
- SQL Create Context Instruction[47] [P2]: This dataset contains 78,577 examples of natural language queries, SQL CREATE TABLE statements that serve as context, and SQL Queries answering the question. This dataset is built upon the SQL Create Context dataset, which was constructed using data from WikiSQL and Spider.
- CommitPackFT[48] [P2]: A filtered version of CommitPack containing only high-quality commit messages that resemble natural language instructions (Muennighoff et al., 2023b).
- OASST-OctoPack[49] [P2]: A filtered version of code data extracted from OASST (Muennighoff et al., 2023b), which only covers high-quality conversation trees.
- FLAN [50] [P2]: A filtered version of the original FLAN dataset, by only keeping permissible license datasets. (Wei et al., 2022).
- WebInstructSub[51] [P2]: A high-quality subset of the MAmmoTH2 dataset (Yue et al., 2024).
- Open-Platypus [52] [P2]: A dataset to improve LLM's logical reasoning skills. We filtered the original dataset to keep only permissible licensed subsets (Lee et al., 2024a).
- xP3x-octopack[53] [P2]: A subset of code-related instances extracted from xP3x, a permissive-license instruction dataset. The extraction process was performed as part of the OctoPack project (Muennighoff et al., 2023b).
- Aya Dataset[54] [P2][ML12]: A multilingual instruction tuning dataset curated by an open-science community which contains a total of 204k human-annotated prompt-completion pairs along with the demographics data of the annotators (Singh et al., 2024).
- Function Calling/API Data [P2][IBM-Synthetic]: Synthetic data covering different types of tool-calling scenarios.
- Reasoning Instructions [P2][IBM-Synthetic]: Synthetic data created using code assistance and knowledge bases to improve reasoning capabilities of Granite models.
- Language Instructions [P2][IBM-Synthetic]: Synthetic dataset of instruction-response pairs created using EvolInstruct to improve complex reasoning and conversation skills.
- Cybersecurity Instructions [P2]: A synthetic dataset of instruction-response pairs about cybersecurity topics, as described in 3.

B.1.6 Academic

- peS2o[55][P1, P2]: The peS2o dataset is a collection of 40M creative open-access academic papers, cleaned, filtered, and formatted for pre-training of language models. It is derived from the Semantic Scholar Open Research Corpus (S2ORC).
- arXiv [P1, P2][IBM-Curated]: Scientific paper pre-prints posted to arXiv. Full author acknowledgement can be found here.
- IEEE [P1, P2][IBM-Curated]: Technical content from IEEE acquired by IBM. Full author acknowledgement can be found here.

[45] https://huggingface.co/datasets/bigcode/self-oss-instruct-sc2-exec-filter-50k
[46] https://huggingface.co/datasets/glaiveai/glaive-code-assistant-v3
[47] https://huggingface.co/datasets/bugdaryan/sql-create-context-instruction
[48] https://huggingface.co/datasets/bigcode/commitpackft
[49] https://huggingface.co/datasets/bigcode/oasst-octopack
[50] https://huggingface.co/datasets/Muennighoff/flan
[51] https://huggingface.co/datasets/TIGER-Lab/WebInstructSub
[52] https://huggingface.co/datasets/garage-bAInd/Open-Platypus
[53] https://huggingface.co/datasets/bigcode/xp3x-octopack
[54] https://huggingface.co/datasets/CohereForAI/aya_dataset
[55] https://huggingface.co/datasets/allenai/peS2o

- DeepMind Mathematics [P1, P2][IBM-Curated]: Mathematical question and answering data designed to test mathematical learning and algebraic reasoning skills of models.
- Financial Research Papers [P1, P2][IBM-Curated]: Publicly available financial research paper corpus, curated and filtered by IBM.
- Papers With Code[56] [P1, P2][IBM-Curated]: Selection of publicly available academic papers sourced from the papers with code website.

B.1.7 TECHNICAL

- Wikipedia [P1, P2][IBM-Curated]: Technical articles sourced from Wikipedia.
- Library of Congress Public Domain Books[57] [P1, P2]: This dataset contains more than 140,000 English books digitised by the Library of Congress (LoC) that are in the public domain in the United States.
- Directory of Open Access Books [P1, P2][IBM-Curated]: Selection of publicly available technical books sourced from the Directory of Open Access Books, a community-driven service that indexes and provides access to scholarly, peer-reviewed open access books.
- Cosmopedia[58][P2]: A dataset of synthetic textbooks, blogposts, stories, posts and WikiHow articles (Ben Allal et al., 2024).

B.1.8 MATH

- OpenWebMath[59] [P1, P2]: This dataset contains the majority of the high-quality, mathematical text from the internet. It is filtered and extracted from over 200B HTML files on Common Crawl down to a set of 6.3 million documents containing a total of 14.7B tokens. We used a filtered version of this dataset.
- Algebraic-Stack[60][P1, P2]: A new dataset of mathematical code composed of 11B tokens that includes numerical computing, computer algebra, and formal mathematics. This dataset is part of the Proof-Pile-2 collection (Paster et al., 2023).
- Stack Exchange [P1, P2][IBM-Curated]: Anonymized set of all user-contributed content on the Stack Exchange network, a popular collection of websites centered around user-contributed questions and answers.
- MetaMathQA[61][P2]: a dataset built by rewriting mathematical questions from multiple perspectives (Yu et al., 2023).
- StackMathQA[62] [P2]: a meticulously curated collection of 2 million mathematical questions and answers, sourced from various Stack Exchange sites (Zhang, 2024).
- MathInstruct[63] [P2]: a synthetic dataset focusing on the hybrid use of chain-of-thought (CoT) and program-of-thought (PoT) rationales, and ensures extensive coverage of diverse mathematical fields (Yue et al., 2023).
- TemplateGSM[64] [P2]: a novel and extensive collection containing over 7 million grade school math problems with code solutions and natural language solutions designed for advancing the study and application of mathematical reasoning within the realm of language modeling and AI (Zhang et al., 2024b).

B.2 POST-TRAINING DATA

This section outlines the datasets employed during post-training of our models including supervised finetuning and alignment. To recognize specific features, we utilize the following annotations:

[56] https://paperswithcode.com/
[57] https://huggingface.co/datasets/storytracer/LoC-PD-Books
[58] https://huggingface.co/datasets/HuggingFaceTB/cosmopedia
[59] https://huggingface.co/datasets/open-web-math/open-web-math
[60] https://huggingface.co/datasets/EleutherAI/proof-pile-2
[61] https://huggingface.co/datasets/meta-math/MetaMathQA
[62] https://huggingface.co/datasets/math-ai/StackMathQA
[63] https://huggingface.co/datasets/TIGER-Lab/MathInstruct
[64] https://huggingface.co/datasets/math-ai/TemplateGSM

- IBM-Synthetic: synthetic data created by IBM.
- ML12: sources from which we selected data in the following languages: English, German, Spanish, French, Japanese, Portuguese, Arabic, Czech, Italian, Korean, Dutch, Chinese.

B.2.1 GENERAL ENGLISH

- Open-Platypus [65]: A dataset to improve LLM's logical reasoning skills. We filtered the original dataset to keep only permissible licensed subsets (Lee et al., 2024a).
- WebInstructSub[66]: A high-quality subset of the MAmmoTH2 dataset (Yue et al., 2024).
- OASST-OctoPack[67]: A filtered version of code data extracted from OASST (Muennighoff et al., 2023b), which only covers high-quality conversation trees.
- Daring-Anteater[68]: A comprehensive dataset for instruction tuning covering a wide range of tasks and scenarios (Wang et al., 2024b).
- SoftAge-Multiturn[69]: A dataset of 400 text-only fine-tuned versions of multi-turn conversations in English based on 10 categories and 19 use cases.
- Glaive-RAG-v1[70]: A dataset with 50k samples built using the Glaive platform, for finetuning models for RAG use cases.
- EvolKit-20k[71]: A 20k synthetic high quality samples dataset build by following the Evol-Instruct method via EvolKit[72] framework.
- Magpie-Phi3-Pro-300K-Filtered[73]: A synthetic high-quality single-turn instruction dataset generated using MagPie with microsoft/Phi-3-medium-128k-instruct.
- HelpSteer2 [74]: A helpfulness dataset that supports aligning models to become more helpful, factually correct and coherent, while being adjustable in terms of the complexity and verbosity of its responses.
- Truthy-DPO [75]: A dataset designed to enhance the overall truthfulness of LLMs, without sacrificing immersion when roleplaying as a human.
- Synthetic ShareGPT Prompts [76] [IBM-Synthetic]: We take the ShareGPT prompts without responses and use an LLM to generate multi-turn data to improve conversational skills.
- Reasoning Instructions [IBM-Synthetic]: Synthetically generated high quality reasoning data to improve the reasoning abilities of Granite models. We use code-assisted synthetic data generation as well as knowledge-based data generation techniques to create this dataset.
- Cybersecurity Instructions [IBM-Synthetic]: A collection of synthetic datasets grounded in security data sources to generate both rules-based and synthetic security instructions by combining Evol-Instruct (Xu et al., 2023) and Self-Instruct Wang et al. (2022) methods alongside content-grounded generation and evaluation pipelines.
- Synthetic LMSys-Chat-1M[77] [IBM-Synthetic]: We take the LMSys-Chat-1M prompts without the responses and use an LLM to generate multi-turn conversation data to better mirror real-world user requests.
- Evol Open-Platypus [IBM-Synthetic]: A synthetic single-turn instruction dataset generated by following the Evol-Instruct method on top of instructions from Open-Platypus[78] dataset. Post-filtering was applied to obtained instances with the highest quality.

[65] https://huggingface.co/datasets/garage-bAInd/Open-Platypus
[66] https://huggingface.co/datasets/TIGER-Lab/WebInstructSub
[67] https://huggingface.co/datasets/bigcode/oasst-octopack
[68] https://huggingface.co/datasets/nvidia/Daring-Anteater
[69] https://huggingface.co/datasets/SoftAge-AI/multi-turn_dataset
[70] https://huggingface.co/datasets/glaiveai/RAG-v1
[71] https://huggingface.co/datasets/arcee-ai/EvolKit-20k
[72] https://github.com/arcee-ai/EvolKit
[73] https://huggingface.co/datasets/Magpie-Align/Magpie-Phi3-Pro-300K-Filtered
[74] https://huggingface.co/datasets/nvidia/HelpSteer2
[75] https://huggingface.co/datasets/jondurbin/truthy-dpo-v0.1
[77] https://huggingface.co/datasets/lmsys/lmsys-chat-1m
[78] https://huggingface.co/datasets/garage-bAInd/Open-Platypus

- MagPie Synthetic [IBM-Synthetic]: We created two synthetic single-turn instruction datasets by following MagPie (Xu et al., 2024) using two teacher language models. Post-filtering was applied to obtain instances with the highest quality from both datasets. We also extended the turns of the filtered version of MagPie-Mistral-Nemo-Instruct-2407-Filtered-Single-Turn dataset to create a multi-turn version to improve conversation skills.
- Synthetic Everyday Conversations [IBM-Synthetic]: Synthetic dataset about simple multi-turn conversations between an user and an AI assistant about a given topic. We use prompts from everyday-conversations-llama3.1-2k [79] and adopt a permissive license language model as the text generation language model to create this dataset.
- Incapable Tasks [IBM-Synthetic]: A synthetic dataset to teach LLMs how to respond to tasks that they are incapable of performing by themselves. We created this dataset by using a permissive license teacher language model as the text generation language model.
- Hardcoded [IBM-Synthetic]: A collection of hardcoded prompts to ensure the model generates correct outputs given inquiries about its name or developers.
- InstructLab Data [80][IBM-Synthetic]: A high-quality synthetic dataset generated using InstructLab's taxonomy driven data generation framework.
- Product Feedback [IBM-Synthetic]: A synthetic dataset curated to address real user concerns from IBM's Watsonx platform.

B.2.2 MULTILINGUAL

- Aya Dataset[81] [ML12]: A multilingual instruction fine-tuning dataset curated by an open-science community via Aya Annotation Platform from Cohere For AI. The dataset contains a total of 204k human-annotated prompt-completion pairs along with the demographics data of the annotators Singh et al. (2024).
- LLM-Japanese-Dataset: A japanese chat dataset for tuning large language models consisting of about 8.4 million records (Hirano et al., 2023).
- Japanese-OASST [82]: A machine translated version of original OASST dataset (Köpf et al.).
- Machine Translation Data [IBM-Curated]: Machine translation datasets from ParaCrawl [83], WikiMatrix (Schwenk et al., 2019a), and NLLB/CCMatrix (Schwenk et al., 2019b).
- Daring Anteater Translated [IBM-Synthetic][ML12]: A multilingual dataset created by translating the Daring-Anteater[84] dataset, a comprehensive instruction dataset covering a wide range of tasks and scenarios, from English to other languages.

B.2.3 CODE

- Glaive Code Assistant V3[85]: A dataset of approximately 1M code problems and solutions generated using Glaive's synthetic data generation platform. This source includes the first and second versions of the dataset.
- SQL Create Context Instruction[86]: This dataset contains 78,577 examples of natural language queries, SQL CREATE TABLE statements that serve as context, and SQL Queries answering the question. This dataset is built upon the SQL Create Context dataset, which was constructed using data from WikiSQL and Spider.
- Self-OSS-Instruct-SC2[87]: A synthetic dataset curated from the Stack V1 pretraining dataset without any human annotations or distilled data from huge and proprietary LLMs.

[79] https://huggingface.co/datasets/HuggingFaceTB/everyday-conversations-llama3.1-2k
[80] https://huggingface.co/datasets/instructlab/InstructLabCommunity
[81] https://huggingface.co/datasets/CohereForAI/aya_dataset
[82] https://huggingface.co/datasets/kunishou/oasst1-89k-ja
[83] http://paracrawl.eu
[84] https://huggingface.co/datasets/nvidia/Daring-Anteater
[85] https://huggingface.co/datasets/glaiveai/glaive-code-assistant-v3
[86] https://huggingface.co/datasets/bugdaryan/sql-create-context-instruction
[87] https://huggingface.co/datasets/bigcode/self-oss-instruct-sc2-exec-filter-50k

- Multi-programming SC-Instruct [IBM-Synthetic]: A synthetic coding dataset created using a modified OSS self-instruct pipeline to 6 coding languages: JavaScript, TypeScript, C, C++, Go, and Python with granite-34b-code-instruct model as the teacher model.
- Multiturn Coding Instructions [IBM-Synthetic]: A high quality multi-turn synthetic dataset curated by incorporating execution output as feedback, as in (Zheng et al., 2024).
- CodeGenPlus [IBM-Synthetic]: Synthetic data for coding tasks like code explanation, docstring, debugging, and pseudocode generation.
- Evol-SC-Instruct [IBM-Synthetic]: A synthetic single-turn instruction dataset generated by following the Evol-Instruct method sourcing instructions from Multi-programming SC-Instruct dataset.
- Evol Multiturn Coding Instructions [IBM-Synthetic]: A synthetic dataset created using Evol-Instruct method with instructions from Multiturn Coding Instructions.
- Evol-Self-OSS-Instruct-SC2 [IBM-Synthetic]: A synthetic coding dataset generated by following the Evol-Instruct on top of Self-OSS-Instruct-SC2.
- Evol Glaive Code Assistant V3 [IBM-Synthetic]: A synthetic single-turn coding dataset generated by using Evol-Instruct on top of instructions from Glaive Code Assistant v3 dataset and using a permissive license language model. Post-filtering was applied to obtain instances with the highest quality.

B.2.4 MATH

- MetaMathQA[88]: A synthetic dataset built by rewriting mathematical questions from multiple perspectives (Yu et al., 2023).
- StackMathQA[89]: A meticulously curated collection of 2 million mathematical questions and answers, sourced from various Stack Exchange sites (Zhang, 2024).
- MathInstruct[90]: A mathematical dataset focusing on the hybrid use of chain-of-thought (CoT) and program-of-thought (PoT) rationales (Yue et al., 2023).

B.2.5 TOOLS

- xlam-function-calling[91]: A synthetic dataset created with APIGen (Liu et al., 2024b), an automated data generation pipeline designed to produce verifiable high-quality datasets for function-calling applications.
- Glaive Function Calling V2[92]: A function calling dataset in real-world scenarios.
- Hermes Function Calling V1[93]: A structured output dataset composed by function-calling conversations, json-mode, agentic json-mode, and structured extraction samples, designed to train LLM models in performing function calls based on natural language instructions.
- Function Calling/API Data [IBM-Synthetic]: A synthetic dataset of 1M samples created to cover different types of tool-calling scenarios, as described in Section 3.

[88] https://huggingface.co/datasets/meta-math/MetaMathQA
[89] https://huggingface.co/datasets/math-ai/StackMathQA
[90] https://huggingface.co/datasets/TIGER-Lab/MathInstruct
[91] https://huggingface.co/datasets/Salesforce/xlam-function-calling-60k
[92] https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2
[93] https://huggingface.co/datasets/NousResearch/hermes-function-calling-v1

B.2.6 SAFETY

- Safety Prompts: We use input prompts from the following datasets: SimpleSafetyTests[94], HarmBench Behaviors[95], Reject[96], AdvBench[97], Do-Not-Answer[98] and MistralGuard[99] to generate synthetic data aligned with our safety taxonomy.
- Anthropic-HH-RLHF [100]: Human preference data about helpfulness and harmlessness that we used in model alignment.
- Internal Safety Data [IBM-Synthetic]: A synthetic dataset created for safety alignment of Granite models as described in Section 3.

B.3 REWARD MODEL(S) TRAINING DATA

B.3.1 MULTI-ASPECT REWARD MODEL

We trained our multi-aspect reward model on HelpSteer2 (https://huggingface.co/datasets/nvidia/HelpSteer2) for two epochs.

B.3.2 BRADLEY TERRY REWARD MODEL

We trained a Mistral-7B-Instruct-v0.2 reward model, with Bradley Terry preference objective, using a mix of gold and synthetic datasets. During training, the proportion of gold data was 20% and the remaining 80% was sampled from synthetically generated preference data. The model was trained for 120k steps with a batch size of 16 and a learning rate of 1e-7.

The following gold preference datasets were used in training:

- HelpSteer2_DPO: a binary preference version of helpsteer 2.0 dataset, https://huggingface.co/datasets/gx-ai-architect/HelpSteer2_DPO containing around 7k samples.
- safetyQA_DPO: a safety preference dataset, containing roughly 50k samples, https://huggingface.co/datasets/AmberYifan/safetyQA_DPO
- truthy-dpo-v0.1: A small dataset (1k samples), aiming at improving truthfulness, especially in the context of human-like self-awareness. https://huggingface.co/datasets/jondurbin/truthy-dpo-v0.1
- anthropic_hh: A dataset of over 161k samples, containing preference pairs geared towards helpful and harmless responses. https://huggingface.co/datasets/Anthropic/hh-rlhf
- Agentic-DPO-V0.1: A small 5k samples dataset designed to improve AI models for agentic processing. https://huggingface.co/datasets/Capx/Agentic-DPO-V0.1

We also generated preference data synthetically using the model-gap technique Naseem et al. (2024), and by perturbing gold datasets to create synthetic negatives:

- We generated 96k samples, where the chosen response is generated from Mixtral-8x22B-Instruct-v0.1 and the rejected from Mistral-7B-Instruct-v0.1
- Another 770k samples were generated, where the chosen response is generated from Mixtral-8x7B-Instruct-v0.1 and the rejected from Mistral-7B-Instruct-v0.1
- We also created around 80k preference pairs by perturbing the gold samples in MathInstruct dataset to create synthetic negatives. In particular, we switched one or more numeric values with a different but close value.

[94] https://huggingface.co/datasets/Bertievidgen/SimpleSafetyTests

[95] https://github.com/centerforaisafety/HarmBench/blob/main/data/behavior_datasets/harmbench_behaviors_text_all.csv

[96] https://github.com/alexandrasouly/strongreject/blob/main/strongreject_dataset/strongreject_dataset.csv

[97] https://huggingface.co/datasets/walledai/AdvBench

[98] https://huggingface.co/datasets/LibrAI/do-not-answer

[99] https://huggingface.co/datasets/natolambert/xstest-v2-copy

[100] https://huggingface.co/datasets/Anthropic/hh-rlhf

REFERENCES

Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Sadhana Kumaravel, Matthew Stallone, Rameswar Panda, Yara Rizk, GP Bhargav, Maxwell Crouse, Chulaka Gunasekara, et al. Granite-function calling model: Introducing function calling abilities via multi-task learning of granular tasks. *arXiv preprint arXiv:2407.00121*, 2024.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Md Tanvirul Alam, Dipkamal Bhushl, Le Nguyen, and Nidhi Rastogi. Ctibench: A benchmark for evaluating llms in cyber threat intelligence. *arXiv preprint arXiv:2406.07599*, 2024.

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the association for computational linguistics*, 7: 597–610, 2019.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Kinjal Basu, Ibrahim Abdelaziz, Subhajit Chaudhury, Soham Dan, Maxwell Crouse, Asim Munawar, Sadhana Kumaravel, Vinod Muthusamy, Pavan Kapanipathi, and Luis A Lastras. Api-blend: A comprehensive corpora for training and benchmarking api llms. *arXiv preprint arXiv:2402.15491*, 2024.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Cosmopedia, 2024. URL `https://huggingface.co/datasets/HuggingFaceTB/cosmopedia`.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3-4):324–345, 1952.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17754–17762, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Yujia Chen, Cuiyun Gao, Muyijie Zhu, Qing Liao, Yong Wang, and Guoai Xu. Apigen: Generative api method recommendation. *arXiv preprint arXiv:2401.15843*, 2024b.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2174–2184, 2018.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL `https://aclanthology.org/N19-1300`.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*, 2023.

Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. Bold: Dataset and metrics for measuring biases in open-ended language generation. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 862–872, 2021.

P Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024. URL `https://arxiv.org/abs/2401.08281`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

Abhimanyu Dubey et al. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

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

Song Feng, Siva Sankalp Patel, Hui Wan, and Sachindra Joshi. Multidoc2dial: Modeling dialogues grounded in multiple documents. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6162–6176, 2021.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL `https://zenodo.org/records/12608602`.

Talia Gershon, Seetharami Seelam, Brian Belgodere, et al. The infrastructure powering ibm's gen ai model development. *arXiv preprint arXiv:2407.05467*, 2024.

Zhen Guo, Adriana Meza Soria, Wei Sun, Yikang Shen, and Rameswar Panda. Api pack: A massive multi-programming language dataset for api call generation. *arXiv preprint arXiv:2402.09615*, 2024.

Alexander Havrilla, Maksym Zhuravinskyi, Duy Phung, Aman Tiwari, Jonathan Tow, Stella Biderman, Quentin Anthony, and Louis Castricato. trlX: A framework for large scale reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8578–8595, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.530. URL https://aclanthology.org/2023.emnlp-main.530.

Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. Linking threat tactics, techniques, and patterns with defensive weaknesses, vulnerabilities and affected platform configurations for cyber hunting. *arXiv preprint arXiv:2010.00533*, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Masanori Hirano, Masahiro Suzuki, and Hiroki Sakaji. llm-japanese-dataset v0: Construction of japanese chat dataset for large language models and its methodology. In *International Conference on Network-Based Information Systems*, pp. 442–454. Springer, 2023.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6384–6392, 2021.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Hannah Rose Kirk, Bertie Vidgen, Paul Röttger, and Scott A Hale. Personalisation within bounds: A risk taxonomy and policy framework for the alignment of large language models with personalised feedback. *arXiv preprint arXiv:2303.05453*, 2023.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. corr, abs/2304.07327, 2023. doi: 10.48550. *arXiv preprint arXiv.2304.07327*.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *Advances in Neural Information Processing Systems*, 35:16203–16220, 2022.

George Kour, Marcel Zalmanovici, Naama Zwerdling, Esther Goldbraich, Ora Nova Fandina, Ateret Anaby-Tavor, Orna Raz, and Eitan Farchi. Unveiling safety vulnerabilities of large language models. *arXiv preprint arXiv:2311.04124*, 2023.

Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling, 2024.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms, 2024a.

Young-Suk Lee, Chulaka Gunasekara, Danish Contractor, Ramón Fernandez Astudillo, and Radu Florian. Multi-document grounded multi-turn synthetic dialog generation, 2024b. URL `https://arxiv.org/abs/2409.11500`.

Matan Levi, Yair Alluouche, Daniel Ohayon, and Anton Puzanov. Cyberpal. ai: Empowering llms with expert-driven cybersecurity instructions. *arXiv preprint arXiv:2408.09304*, 2024.

Guancheng Li, Yifeng Li, Wang Guannan, Haoyu Yang, and Yang Yu. Seceval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models. https://github.com/XuanwuAI/SecEval, 2023a.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2024a.

Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 2023b.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. *arXiv preprint arXiv:2402.05044*, 2024b.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, December 2023c. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL `https://aclanthology.org/2023.emnlp-main.187`.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023d.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training, 2020. URL https://arxiv.org/abs/2006.15704.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12286–12312, Toronto, Canada, July 2023e. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.687. URL https://aclanthology.org/2023.acl-long.687.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*, 2024b.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, et al. Granite code models: A family of open foundation models for code intelligence. *arXiv preprint arXiv:2405.04324*, 2024.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*, 2023a.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*, 2023b.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R Bowman. Crows-pairs: A challenge dataset for measuring social biases in masked language models. *arXiv preprint arXiv:2010.00133*, 2020.

Deepak Narayanan, Amar Phanishayee, Kaiyu Shi, Xie Chen, and Matei Zaharia. Memory-efficient pipeline-parallel dnn training. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7937–7947. PMLR, 18–24 Jul 2021a. URL https://proceedings.mlr.press/v139/narayanan21a.html.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021b. URL https://arxiv.org/abs/2104.04473.

Tahira Naseem, Guangxuan Xu, Sarathkrishna Swaminathan, Asaf Yehudai, Subhajit Chaudhury, Radu Florian, Ramón Astudillo, and Asim Munawar. A grounded preference model for LLM alignment. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 151–162, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.10. URL https://aclanthology.org/2024.findings-acl.10.

Yonatan Oren, Shiori Sagawa, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust language modeling. *arXiv preprint arXiv:1909.02060*, 2019.

Gaurav Pandey, Yatin Nandwani, Tahira Naseem, Mayank Mishra, Guangxuan Xu, Dinesh Raghu, Sachindra Joshi, Asim Munawar, and Ramón Fernandez Astudillo. BRAIn: Bayesian reward-conditioned amortized inference for natural language generation from feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39400–39415. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/pandey24a.html.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 2024.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Sujoy Roychowdhury, Sumit Soman, HG Ranjani, Neeraj Gunda, Vansh Chhabra, and Sai Krishna Bala. Evaluation of rag metrics for question answering in the telecom domain. *CoRR*, 2024.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. *arXiv preprint arXiv:1907.05791*, 2019a.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. Ccmatrix: Mining billions of high-quality parallel sentences on the web. *arXiv preprint arXiv:1911.04944*, 2019b.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, et al. Bond: Aligning llms with best-of-n distillation. *arXiv preprint arXiv:2407.14622*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy Zhang, Sahil Jain, Ali Taghibakhshi, Markel Sanz Ausin, Ashwath Aithal, and Oleksii Kuchaiev. Nemo-aligner: Scalable toolkit for efficient model alignment, 2024a.

Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*, 2024b.

Yikang Shen, Matthew Stallone, Mayank Mishra, Gaoyuan Zhang, Shawn Tan, Aditya Prasad, Adriana Meza Soria, David D Cox, and Rameswar Panda. Power scheduler: A batch size and token number agnostic learning rate scheduler. *arXiv preprint arXiv:2408.13359*, 2024c.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL `https://arxiv.org/abs/1909.08053`.

Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Minh Chien, Sebastian Ruder, Surya Guthikonda, Emad A. Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. Aya dataset: An open-access collection for multilingual instruction tuning, 2024.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning, 2024. *URL https://arxiv. org/abs/2310.16049*.

Venkat Krishna Srinivasan, Zhen Dong, Banghua Zhu, Brian Yu, Damon Mosk-Aoyama, Kurt Keutzer, Jiantao Jiao, and Jian Zhang. Nexusraven: a commercially-permissive language model for function calling. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Shawn Tan, Yikang Shen, Rameswar Panda, and Aaron Courville. Scattered mixture-of-experts implementation. *arXiv preprint arXiv:2403.08245*, 2024.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Simone Tedeschi, Felix Friedrich, Patrick Schramowski, Kristian Kersting, Roberto Navigli, Huu Nguyen, and Bo Li. Alert: A comprehensive benchmark for assessing large language models' safety through red teaming. *arXiv preprint arXiv:2404.08676*, 2024.

Norbert Tihanyi, Mohamed Amine Ferrag, Ridhi Jain, and Merouane Debbah. Cybermetric: A benchmark dataset for evaluating large language models knowledge in cybersecurity. *arXiv preprint arXiv:2402.07688*, 2024.

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

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024a.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models, 2024b. URL https://arxiv.org/abs/2406.08673.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022. URL https://arxiv.org/abs/2109.01652.

David Wood, Boris Lublinsky, Alexy Roytman, Shivdeep Singh, Abdulhamid Adebayo, Revital Eres, Mohammad Nassar, Hima Patel, Yousaf Shah, Constantin Adam, Petros Zerfos, Nirmit Desai, Daiki Tsuzuku, Takuya Goto, Michele Dolfi, Saptha Surendran, Paramesvaran Selvam, Sungeun An, Yuan Chi Chang, Dhiraj Joshi, Hajar Emami-Gohari, Xuan-Hong Dang, Yan Koyfman, and Shahrokh Daijavad. Data-prep-kit: getting your data ready for llm application development, 2024. URL https://arxiv.org/abs/2409.18164.

Mengsong Wu, Tong Zhu, Han Han, Chuanyuan Tan, Xiang Zhang, and Wenliang Chen. Seal-tools: Self-instruct tool learning dataset for agent tuning and detailed benchmark. *arXiv preprint arXiv:2405.08355*, 2024.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing, 2024. URL https://arxiv.org/abs/2406.08464.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. Paws-x: A cross-lingual adversarial dataset for paraphrase identification. *arXiv preprint arXiv:1908.11828*, 2019.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models, 2023.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Yu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*, 2024.

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

Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. Llm as a mastermind: A survey of strategic reasoning with large language models. *arXiv preprint arXiv:2404.01230*, 2024a.

Yifan Zhang. Stackmathqa: A curated collection of 2 million mathematical questions and answers sourced from stack exchange, 2024.

Yifan Zhang, Yifan Luo, Yang Yuan, and Andrew Chi-Chih Yao. Training language models with syntactic data generation, 2024b.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement, 2024.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2299–2314, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.149. URL https://aclanthology.org/2024.findings-naacl.149.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.