INF's Open-Source Large Language Models

INF-Team

July 19, 2024

This technical report is a companion document to our white paper "Towards Trustworthy Large Language Models in Industry Domains" [1], and complements the full stack technology of large language models. We release a large language model to the open source community, named INF-34B, under an INF license that is friendly to research and commercial use. INF-34B has 34 billion parameters with a context window length of 32K and is trained on about 3.5T well-processed tokens from our curated English and Chinese bilingual corpus. In the report we present training details and report the results of model evaluation on widely used benchmarks. Compared with open-source models of comparable size, INF-34B not only provides competitive performance in the OpenCompass evaluation, but also has impressive potential in both the finance and healthcare domains. In addition to its outstanding comprehensive capacities, the quantized INF-34B runs on low-resource graphics cards with negligible accuracy loss, which helps it be suitable for low-resource applications.

1. Introduction

Large language models (LLMs) have attracted widespread attention from both research and industry communities, since OpenAI launched ChatGPT in November 2022. This breakthrough has been widely applied to chatbots for customer service, language translation, coding copilot, and creative writing. Natural language becomes a direct interactive interface between humans and machines.

Inspired by the scaling law of LLMs [2], recent efforts have focused on improving model accuracy by pretraining much larger models in terms of parameters and training data, such as Llama-3 family trained on 15 trillion tokens [3], and the Nemotron-4 340B trained with 9 trillion tokens [4]. Though the principle still applies that a larger model achieves better general performance, it is critical to trade-

off between deployment overhead and model performance in practical applications. The hardware requirement for deploying the Nemotron-4 340B model is 16xA100 80GB, i.e. two nodes of A100 80GB, as recommended by [4], while deploying the Llama 3 70B model requires at least 2xA100 80GB. In terms of return on investment, a compact model of competitive performance is preferred.

In many scenarios, we only need an expert in a specific domain rather than an all-around generalist. For instance, financial analysts lack of expertise in medicine and biology, which does not impede their performance in analyzing financial data to make investment recommendations. If we are looking for deep expertise in a specialized field, it is feasible to do continuous training on a strong pre-trained model to enhance domain knowledge in order to gain superior performance.

To support the continued adoption of low-resource LLMs or domain-specific LLMs across the open source community, we release INF-34B-Base and INF-34B-Chat as open-source models with a permissive license for both research and commercial use.

We select the public OpenCompass [5] for model evaluation to ease the reproducibility of the metrics. Both the base and chat models are evaluated on multiple dimensions, such as general knowledge, reasoning, math, and coding. We also test the chat model on the instruction-following and long-context ability. To evaluate model potentials in industry domains, we run tests on two domain-specific datasets CFA and USMEL for finance and healthcare respectively. Figure 1 highlights the accuracy of the INF-34B models across selected tasks. We chose two open-source models of similar size as comparison baselines. As shown in Figure 1(a), our base model INF-34B-Base is competitive with the baseline models on reasoning tasks like BBH and HellaSwag, mathematics tasks like GSM8K and MATH, and coding tasks like HumanEval and MBPP. In both mathematics and coding tasks, INF-34B-Base shows very strong performance, which is welcome in many real-word applications, such as financial analysis. In Figure 1(b), the chat model INF-34B-Chat yields competitive results on four instruction-following benchmarks, including MT-Bench, AlignBench, IFEval and Arena Hard, and a superior score on average on the LongBench benchmark in long context evaluation. In Figure 1(c), INF-34B-Chat achieve comparable performance with the double-sized model on the two domain-specific test sets, CFA and USMEL.

To further improve domain capabilities, it is encouraged to conduct domain-specific continuous training [1]. More importantly, we propose to integrate neural symbolic systems with LLMs, leveraging LLMs for fast "black-box" probabilistic predictions while also supporting "white-box" logical reasoning. This integration provides a "gray-box" approach to developing trustworthy LLMs for industrial applications. In Figure 1(c), the scores of INF-trustworthy, our proprietary trustworthy LLMs, quoted from [1], show a significant improvement over the strong baseline models. For more information on domain-specific data preparation and continuous model training, see [1].

Overall, our open-source models deliver competitive performance in all evaluations and show impressive potential in both the finance and healthcare domains. We also report the performance of the quantized chat model in Appendix G, and find that it is comparable in accuracy to the original chat model, but requires one-third less VRAM for deployment and fits on a GeForce RTX™ 4090 GPU. We believe that INF-34B strikes a good balance between performance and cost and can serve as a general-purpose solution for low-resource applications today.

Below is a summary of our contributions.

- We provide comprehensive details about our model pretraining and alignment, including high-quality data pipeline, instruction data preparation, and quantization results etc.
- We demonstrate superior performance of the INF-34B models on the public OpenCompass benchmarks by comparing against two competitive open-access LLMs of comparable model

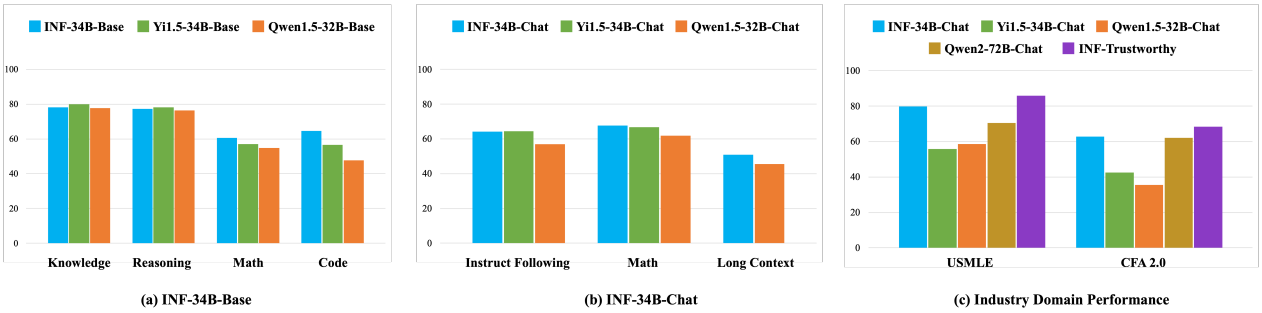

Figure 1 | Result comparison of INF-34B-Base and INF-34B-Chat. See evaluation details in Section 2.3, Section 3.3.1 & Section 3.3.3, and Section 3.3.2, respectively. In (a), the average of MMLU and CMMLU is used for the score of knowledge, the average of BBH and HellaSwag for Reasoning, the average of GSM8K and MATH for Math, and the average of HumanEval and MBPP for Coding, as in Table 3. In (b), the score of Instruct Following is the average of MT-Bench, AlignBench, IFEval and Arena Hard, as in Table 5 where the scores of MT-Bench and AlignBench are converted to 100 scale. The score of Long Context is the average score of all tasks as in Table 7. In (c) the scores are mainly from Table 6 and the scores of INF-Trustworthy are of our proprietary LLMs cited from [1].

size.[1]

- We release the INF-34B models, including INF-34B-Base and INF-34B-Chat, under a permissive license to facilitate commercial applications, especially low-resource scenarios.[2]

The remaining of the report is organized as follows. In Section 2, we introduce pretraining details and evaluate the resulting INF-34B-Base. In Section 3, we report alignment details including instruct tuning, RLHF, safety and long-context, along with the evaluation results of INF-34B-Chat. In Section 4, we conclude and discuss future work.

2. Pretraining

2.1. Data Pipelines

This subsection details our data-cleaning pipeline, including the general, domain, and code data pipelines. The general data pipeline involves general processing methods. For the domain data of interest, e.g., math, wiki, code, we propose a domain-specific data pipeline to extract the domain data from Common Crawl (CC). We also devise a code-specific pipeline to handle massive code data, since the code data has proven its effectiveness in improving the model's reasoning and comprehension ability [6]. Using these pipelines, we obtained 3.5TB tokens of training data in Chinese, English, and code.

2.1.1. General Data Pipeline

Our text cleaning pipeline mainly includes two stages: filtering and deduplication. Despite the similarity with [7, 8], our aim is to reveal the details behind our pipeline, especially the details for Chinese. The filtering involves language identification, URL filtering, and heuristic filtering rules. The deduplication includes both fuzzy deduplication and exact deduplication techniques. The overall process is illustrated in Fig. 2.

[1]The scripts to reproduce the evaluation results can be found at *https://github.com/infly-ai/INF-LLM/*.

[2]The model weight files can be accessed at *https://huggingface.co/infly/*.

Filtering To ensure the quality and relevance of the training data for our model, we implemented a comprehensive set of filtering rules tailored to both English and Chinese texts, based on the Redpajama V2 [9]. These rules were designed to eliminate noise, redundant information, and low-quality samples that could compromise performance. Filtering criteria were meticulously designed to account for the linguistic differences of each language and the specific characteristics of each dataset. We show our default filters below, whose thresholds are adjusted for high filter precision in the CC datasets. For other datasets, we developed an annotation tool to efficiently curate specific settings. By applying these comprehensive filtering rules, the low-quality data were likely excluded in our datasets. We attach the rules in the App. A.

Deduplication Deduplication includes fuzzy deduplication and exact deduplication[10]. For fuzzy deduplication, we used minhash [11] and locality-sensitive hashing (LSH) [12]. The process involves the following steps. To standardize the text, we remove punctuation, lowercase the text, and convert it into the NFD Unicode style [3]. Then we tokenize the text using Jieba tokenizer [4] to adapt Chinese, followed by applying 5-gram processing. We then use the 5-gram pieces to compute the 2048 minhash values for each text. To compute efficiently, we employ a cluster of 20K CPU cores and carefully reduce memory usage by elaborating the calculation steps. We used an LSH setting, where 2048 minhash values are split into 128 bands and 16 rows. We only keep one sample over the samples colliding in any bands. Therefore, we can approximately deduplicate a pair in 80% Jaccard similarity with a high probability of 99.7%. With these improvements, we can perform fuzzy deduplication for a 10TB disk size of data in one hour.

For the exact deduplication stage, we used the suffix array algorithm [13], based on the project [14]. More specifically, we also performed the same text standardization, followed by tokenization using our tokenizer (Sect. 2.2.1). We then used the suffix array algorithm to detect duplicated substrings with 50+ tokens, where we used at least 20 workstations with 2TB of memory to maximize the size of each data split per run. Note that we truncate the tokens into 16 bits to save the storage. Then we remove the documents that contain more than 20% duplicating tokens as in [15]. From our experimental results, we obtained evident improvement by using exact deduplication.

2.1.2. Specific Domain Data Retrieval

We propose an iterative, high-quality data retrieval method based on existing techniques [7, 16, 17], which retrieves relevant data from the Common Crawl (CC) dataset for various target domains, used in stage 3 training.

Code-like and Math-like Data Collection Despite the large amount of English code data, the Chinese code data is relatively short, which hinder the coding ability in Chinese. To remedy this problem, we adopted a similar method as in DeepSeekMath [16] to collect the code-like Chinese data. Specifically, we firstly collect Chinese markdown data from GitHub and use the autonomous data selection method [18] to filter and select 50,000 samples. With these samples as positive samples and an equal number of negative documents randomly selected from CC, we train a classifier using fastText with a vector dimension of 256, a learning rate of 0.1, a maximum n-gram length of 3, and a maximum word occurrence of 3 for 3 epochs. Then we use the trained classifier to label all the CC documents and group them by their domains (root URL, e.g., *www.google.com*). The domains containing more than 10% positive documents are regarded as code-like candidates. Finally, we manual annotate

[3] https://en.wikipedia.org/wiki/Unicode_equivalence

[4] https://github.com/fxsjy/jieba

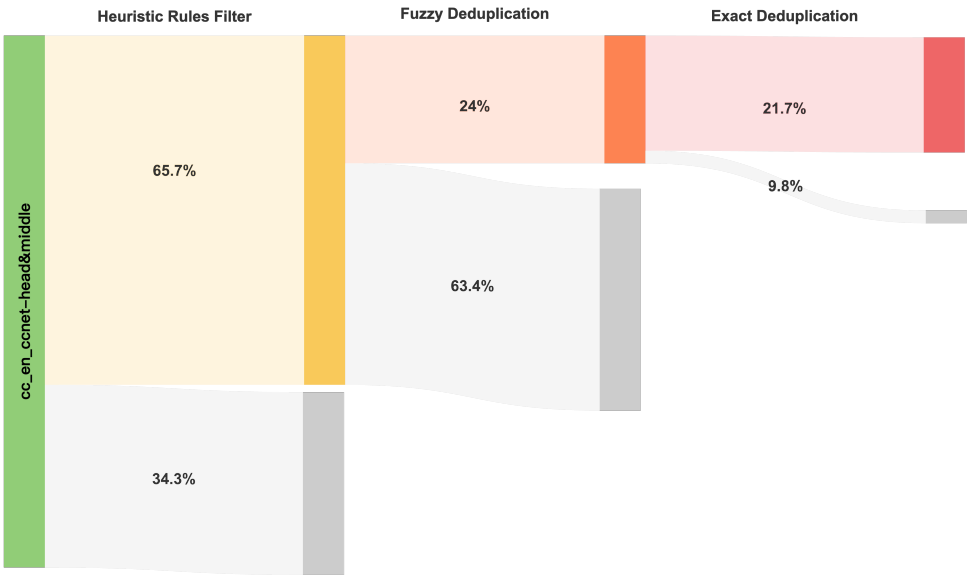

Figure 2 | The figure illustrates the processing workflow and filtering procedures for Common Crawl English data within our pipeline.

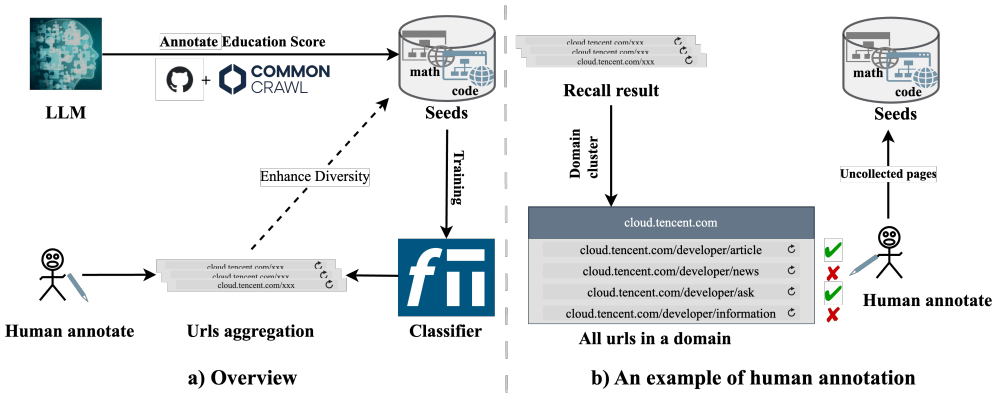

Figure 3 | The figure illustrates the processing of high-quality code-like and math-like recall. The framework is illustrated on the left (a) with the annotation process (b).

these domains and refine the training data. Following our pipeline, we successfully identify the domains that were not correctly classified by fastText, e.g., *cloud.tencent.com/developers/article*, into our corpus. We perform several iterations to enrich and calibrate our datasets. After three iterations, we obtain a collection of 72GB of Chinese code-like data. The distribution of the top websites in code-like data is shown in Fig. 4 and detailed in App. B.1. A similar pipeline was used to select 20.6M math-like document in Chinese (see details in App. B.2).

Wiki-like Data Collection Since the Wiki pages contains rich educative information, we also gather the Wiki-like data from CC. The specific process is shown in Figure 5. We select 17% of Baidu-baike data using filter rules as the seed data, which is used to train a fastText. FastText is used to filter all CC samples whose URL contains "wiki". Among these samples, 200,000 of them are then annotated by GPT-4 for training powerful TinyBERT-4L [19] as a binary educational-level classifier. We apply this classifier to annotate all filtered samples, then use the identified educational samples to retrain a fastText model. This fastText is more robust in detecting Wiki-like data from diverse CC samples.

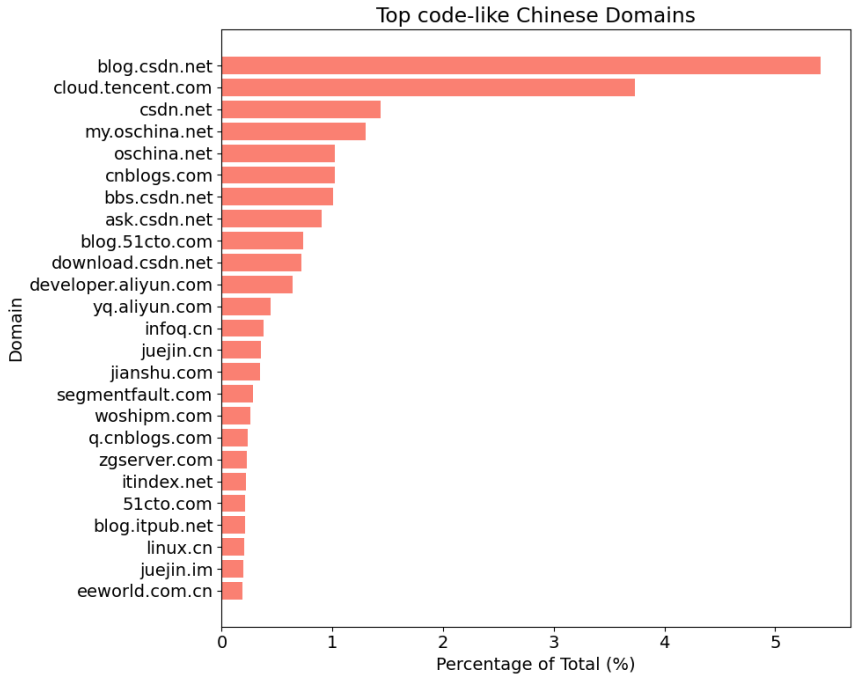

Figure 4 | The distribution of the top domains in code-like data

More details can be found in App. B.3. Ultimately, we recall a total of 6.6 million high-quality Wiki-like Chinese samples from CC.

2.1.3. Code Data Pipeline

To enhance the coding capabilities of the large language model, we carefully develop our code-specific data pipeline to extract useful code data, which incorporates a substantial amount of source code text into the training corpus, from all GitHub repositories up to November 2023. Based on general data pipeline, the code pipeline includes processes of pre-processing, heuristic filtering, and deduplication. The implementation details are displayed in App. C.

2.2. Model Training Details

2.2.1. Tokenizer

We use SentencePiece BPE with byte-level fallback as our tokenizer. During pre-tokenization, we split numbers into 1 to 3 digits following GPT series[20]. For English tokens, we directly copy the first 65536 tokens from the GPT4 (cl100k_base) vocabulary and keep those in the languages of interest. Then we train a tokenizer with vocabulary size of 30000 on a 25GB Chinese corpus sampling from various data source to extract Chinese tokens[21]. We drop tokens that have more than 5 Chinese characters as such tokens usually come from artifacts of a low-quality dataset or the signature of websites. Finally, we add common unicode characters in web data such as emoji to further enhance the compression rate. The resulting tokenizer has nearly 96536 tokens with decent compression rate for English, Chinese and Code.

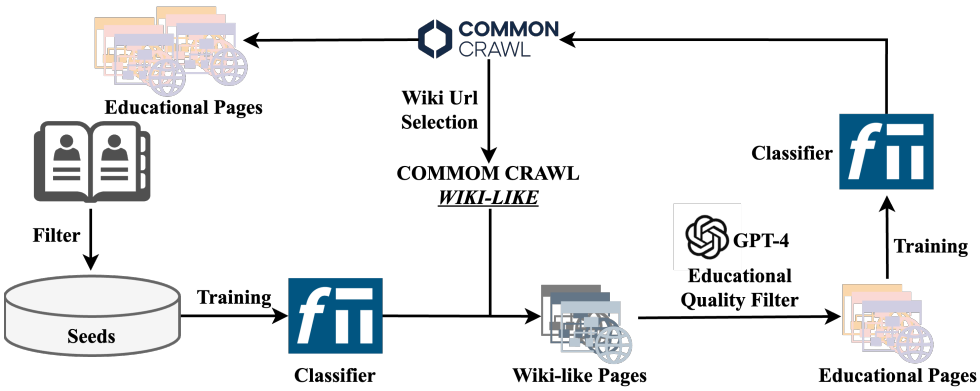

Figure 5 | The figure illustrates the processing of high-quality Wiki-like recall.

2.2.2. Staged Pretraining

We train a 34B model from scratch with the purpose of studying the learning dynamics and demonstrating the effectiveness of our data pipeline and training framework. The architecture choices of INF-34B follows common practice of open-source Llama family [22]. Specifically, we opt Rotary Embedding for positional encoding[23], SwiGLU for activation function, Grouped Query Attention (GQA) and Layernorm with zero-centered gamma instead of RMSNorm for training stability.

Motivated by the idea of first training on a relatively large but less polished corpus to equip the model with language understanding and world knowledge and then improving the model's domain knowledge and reasoning ability[24–26], our training process is split into 3 stages:

- **Stage 1**: The dataset mainly includes the curated web text and source code generated by our data pipeline introduced in sections 2.1.1 and 2.1.3. Arxiv paper and Wikipedia data is also added. In this early stage, we aim at larger data and higher diversity.
- **Stage 2**: For the second stage, we seek to gradually challenge the model with longer and more complex texts. We up-weight long texts in the same data distribution of stage 1. We tune the rope frequency and extend our context window to 32k with the belief that being capable of utilizing information within larger context is crucial for the model to develop more sophisticated comprehension of human knowledge.
- **Stage 3**: The final stage is composed of domain data recalled from Web text and synthetic data. As introduced in section 2.1.2, we train Bert models to recall data related to different targeted domains from CC and then employ a previously trained chat model to rewrite the recalled web text into different formats such as question-answering or Wiki articles. We name these data with the suffix "CC-recall". In addition, similar to recent works[27, 28], we add a small portion of open-source synthetic question-answering style alignment examples to facilitate downstream evaluations and alignment. We perform 13-gram overlap detection against test cases to decontaminate the data in this stage. By preliminary experiments we found this shift of data distribution before alignment would enhance overall performance.

For each of the stages, we use a cosine decay scheduler which warms up 1000 steps to the peak learning rate and then decays to the final learning rate. The detailed hyper parameters and data mixture can be found in Table 1 and Table 2.

	Tokens	batch-size	learning rate	rope-theta	context-len
stage1	2T	8M	2.25e-4 ~ 1e-4	10k	4k
stage2	1T	8M	1e-4 ~ 1.5e-5	500k	32k
stage3	500B	8M	5e-5 ~ 5e-6	500k	4k

Table 1 | Training process and hyperparameters

Category	Datasets	Percentage
NL Pre-train	Common Crawl, Wikipedia Arxiv, UltraTextbook	57%
NL Instructions	QA-CC-recall, OpenHermes[29]	10%
Code	Github-source code Starcoder V2[30], Code-CC-recall Magicoder-OSS, Magicoder-Evol[31]	23%
Math	OpenWebMath[32], Math-CC-recall MetaMathQA[33], MuggleMath[34]	10%

Table 2 | Detailed mixture for final stage

2.2.3. Infrastructures

Our training cluster consists of 64 NVIDIA HGX H800 nodes and each node has 8 H800-80G PCIe GPUs. We maintained an internal version of Megatron-LM-like training framework which compatible with Megatron-Core[35], TransformerEngine, Flash Attention[36], Huggingface[37], etc.. And we also deeply optimized the usability and efficiency for rapidly training and evaluation of Pretrain, Reinforcement Learning and Supervised Fine-tuning.

For INF-34B model training, we employ fundamental tensor parallelism and interleaved pipeline parallelism strategies[35], leveraging ZeRO[38] shared data parallelism with the gradient reduce-scatter and parameter all-gather overlapping separately during the last and first steps of gradient accumulation. We implement uneven bucket splitting of parameters to further enhance the overlap ratio between communication and computation. To further accelerate training without any loss of model accuracy, we pre-compute the rotary embedding matrix and compile the operation of applying the rotary matrix to queries and keys with CUDAGraphs. We further optimize GPU memory by accelerating the freeing of tensors occupied by NCCL tasks and increasing the hit rate of the CUDA memory cache. Eventually, we could reach the peak of 500 TFlops per GPU node while training with BF16 on 512x H800-80G PCIe GPUs.

For 32K long context further pretraining, we incorporate context parallelism with distributed flash attention equipped with load balancing and communication / computation overlapping. We also employ an adaptive gradient checkpointing strategy to maximize resource utilization while maintaining optimal 4D parallelism dimensions.

In training scenarios with non-causal loss mask and packed sequences, we use global token loss averaging to ensure equal treatment for each token, thereby preventing training instability caused by an imbalanced question/answer ratio. Additionally, we adapt to use the DeepSpeed-Ulysses[39] context parallel for about 20%+ better throughput on account of the inefficiency of distributed flash attention for packed long sequences without a standard causal mask.

2.3. Base Model Evaluation

Results In this section, we evaluate our model on several academic benchmarks, and then compare with other similar-sized open-access models. Selected tasks and their category are listed as follows:

- Commonsense reasoning: HellaSwag (0-shot) [40] BBH(3-shot) [41]
- Popular aggregated benchmarks: MMLU (5-shot) [42] CMMLU(5-shot) [43]
- Math related: GSM8K(4-shot) [44] , MATH(4-shot) [45]
- Code: Pass@1 scores on HumanEval (0-shot) [46], Pass@1 scores on MBPP (3-shot) [47]

Although most of our training data are public or synthetic, Table 3 shows that INF-34B maintains a good balance between the general capabilities of LLM such as common sense, world knowledge, mathematics, and coding. We attribute this effectiveness to our data pipeline, which generates information-dense data while still remaining natural and diverse in style. See Appendix H for broader evaluation results.

Model	MMLU	CMMLU	GSM8K	MATH	HumanEval	MBPP	BBH	Hellaswag
Qwen1.5-32B	73.60	81.87	72.86	36.80	44.51	51.00	70.60	82.03
Yi1.5-34B	77.86	81.85	80.06	33.88	47.56	65.60	74.83	81.57
INF-34B	76.11	80.08	83.02	38.34	65.24	64.00	71.20	83.32

Table 3 | Results of the base models on commonly used benchmarks.[1]

Discussion On the journey of pretraining INF-34B model, we are intrigued by several unanswered questions. Despite thorough investigation is mission impossible, preliminary understanding on these questions provides good guidance for practice. In this section, we introduce some of our practical decisions and how we made them.

- **Instruction Data in Stage 3:** Note that INF-34B and several recent works [27, 28] choose to mix benchmark-style instructions in the last pre-training phase, but whether this practice is beneficial or just overdrafting earnings of the alignment stage remains a mystery. We conduct preliminary experiment on a 1.5B model trained on 500B code with the belief that the domain of code can be relatively more controlled for ablation. For both runs, we conduct continued training first, then SFT for 4 epochs, evaluating after each epoch, and report the best result. A slightly better result is observed when the 50% code instructions are consumed in continued training than when training all of these data in the SFT phase, as illustrated in Table 4. We speculate this improvement is attributed to less drastic shift of data distribution between pretraining and instruction-tuning. Notably, alignment data preparation usually requires extensive quality control thus adding unpolished instructions in this stage is cost-effective.
- **Training in FP8:** The introduction of 8-bit floating point (FP8) precision with NVIDIA Hopper GPUs marks a significant advancement, offering enhanced performance while maintaining similar or reduced memory utilization during both the training and inference phases. Additionally, NVIDIA has released a Python library named Transformer Engine [5], which provides a suite of highly optimized building blocks for Transformer architectures. By integrating the Transformer

[1] To facilitate reproduction, the results of common benchmarks are generated by Opencompass [5] except coding tests. Humaneval and MBPP are evaluated using https://github.com/deepseek-ai/DeepSeek-Coder/tree/main/Evaluation

[5] TransformerEngine: https://github.com/NVIDIA/TransformerEngine

Model	Continued Training	SFT	humaneval	mbpp
1.5B	10B source code No instructions	50M instructions*4	37.80	35.00
1.5B-warmup	10B source code 25M instructions*4	25M instructions*4	42.07	38.60

Table 4 | Impact of Instruction Warm-up.

Engine into our training framework, we observed throughput improvements of approximately 30% to 50% compared to BF16 precision training.
Due to the limited practical guidance on the optimal use of FP8 training, the following effort is taken during our training process: We developed a conversion tool to seamlessly switch between the BF16 and FP8 training states, allowing flexible utilization of FP8 training as needed. Our strategy involves enabling FP8 in the early stages of training when precise gradient direction estimation is less critical. We continuously monitor training dynamics to determine appropriate switch points. We observed a moderate spike in the loss curve during the first iteration when transitioning to FP8 training from a BF16 checkpoint. This phenomenon might be due to the improper initialization of the FP8 states. To mitigate this, we use the samples from the first batch to "warm up" the FP8 states, resulting in a smooth and stable loss curve thereafter, as illustrated in Figure 6. When we observed the instability in training, we would switch to BF16 training to address this issue. Figure 7 illustrates this case.

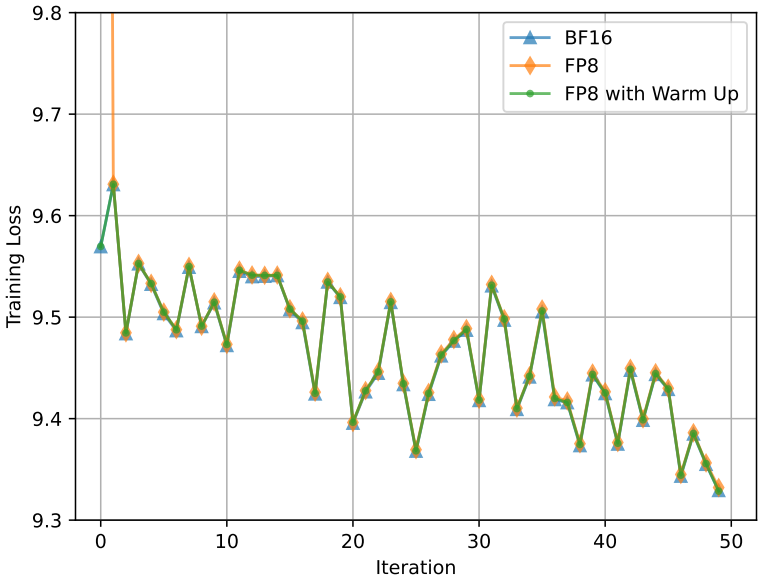

Figure 6 | Loss curves under various training precision and initialization schemes.

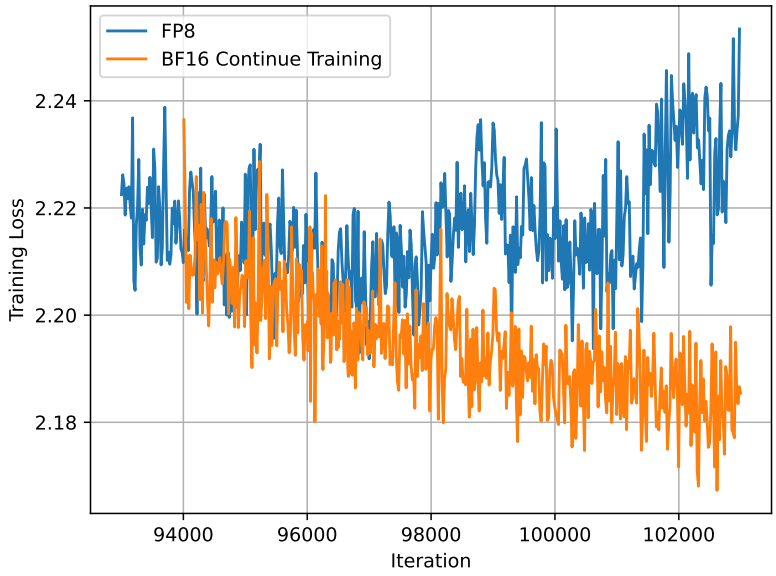

Figure 7 | Training loss curves from 93k to 103k iterations, with different precision settings. BF16 continue training resumes from the FP8 checkpoint at 94k iteration.

3. Alignment

3.1. Supervised Fine-tuning

We initiate the model training with Supervised Fine-Tuning (SFT), which equips the model with fundamental capabilities for chat-like interactions and following instructions, setting the stage for its adaptation to various domain-specific applications. The effectiveness of SFT hinges on three critical factors: the quality of the data, the strategy for data partitioning, and the optimization of hyperparameters. These findings are also supported by several literature [48–54] . High-quality instructional data is pivotal, as it not only enhances the model's response quality but also substantially reduces hallucination. Strategic data partitioning allows us to balance task-specific performance and overall chat functionality, an essential consideration for models expected to operate across different domains. Additionally, the tuning of hyperparameters also plays a crucial role in achieving robust generalization performance. After the whole SFT pipline, we can provide a comprehensive chat-based LLM with known instruction data distribution for subsequent alignment using reinforcement learning. We leave the details of methods for creating high-quality prompts in Appendix D, and the ChatML template specification in Appendix E.

In our preliminary studies, we observed that the acquisition of reasoning capability is more challenging than the acquisition of other capabilities, but reasoning is orthogonal to other capabilities, i.e. augmenting the dataset with a substantial proportion of reasoning-related instructions did not adversely affect the performance on other capacities. Therefore, we strategically incorporated a mixture ratio approaching fifty percent for reasoning instructions and preparing more reasoning related instruction data for last stage of pretraining introduced in Section 2.2.2.

To facilitate the updates of the SFT dataset and ensure that each update progressively enhances the target benchmarks without negatively impacting the capabilities in other domains, we have reorganized and restructured 12 comprehensive categories to represent the model's abilities. The

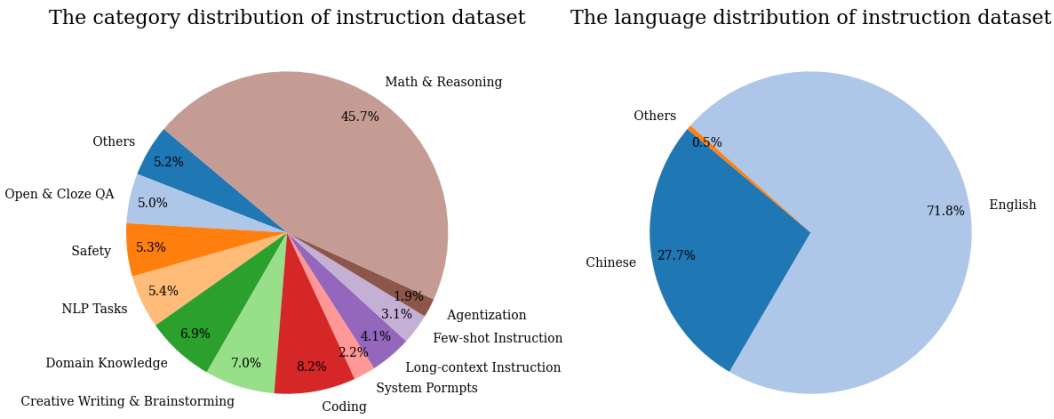

Figure 8 | The data distribution overview of instruction dataset for SFT.

categorical and lingual proportions of the instruction dataset used for INF-34B are illustrated in Figure 8.

3.1.1. Long Context

With an increasing number of large models supporting a context length of up to 32k tokens, the potential applications of these models in tasks such as long document comprehension, content creation, code completion, and specialized domain content analysis have expanded significantly. However, training models with such extensive context windows requires the gathering of substantial amounts of high-quality annotated data, which presents considerable challenges and resource demands.

When constructing SFT datasets for long context, several critical factors should be taken into consideration. In terms of the data sources and preprocessing of data, careful selection of sources can ensure they are diverse and representative of various domains and styles; systematical preprocessing of raw data is paramount to the success of construct supervised finetuning dataset of high quality. Diversity in tasks represented within the dataset is essential to train models capable of addressing various natural language understanding and generation tasks. Furthermore, the distribution of text lengths within the dataset should reflect real-world scenarios to effectively train models for handling different document lengths.

Leveraging large language models with extended context capabilities for tasks involving long texts requires meticulous planning in data sourcing, raw data quality, length distribution, task diversity. Details of our long context data sources and task types can be found in Appendix F. The length distribution of our long context SFT dataset is as follows: 37.7% of the dataset comprises documents shorter than 8k tokens, while 40.5% fall within the range of 8k to 16k tokens. The remaining 21.8% of the dataset consists of documents ranging from 16k to 32k tokens in length. This distribution of document lengths provides a comprehensive sampling across various sizes, ensuring robust training and evaluation in our SFT framework.

3.1.2. Safety

To improve the security of LLMs and better align them with human values, providing users with safe and accurate information, we have implemented a series of measures throughout the model training process. These measures include the development of a comprehensive safety taxonomy, the detoxification of corpus, safety alignment, and rigorous safety evaluations. These efforts collectively

ensure that the final model adheres to the highest standards of safety and reliability.

In the SFT stage, we integrated nearly 25K safety related question-answer pairs into the SFT datas. By adjusting the data ratios and training strategies, we ensured that the introduction of safety data enhances the model's safety without negatively impacting its general capabilities. Using the model trained in SFT stage, we sampled multiple responses to the prepared safety questions and manually scored each response for safety and helpfulness. We also created pairwise comparison data, manually annotating which response was safer and more helpful. This data was then used to train our safety reward model. During the next reinforcement learning (RL) phase, refer to Section 3.2, we employed both safety rewards and helpfulness rewards to increase the likelihood that the model generates safer and more helpful responses.

For risky questions, refusing to answer is always the safest option. However, excessive refusals can significantly degrade user experience and noticeably impact the model's general capabilities. Therefore, when evaluating the overall safety performance of the model, it is essential to balance the risk rate and the negative impact on general capability. Our safety evaluation focusing on both the overall risk rate and the unreasonable refuse rate of the model. The risk rate refers to the proportion of responses containing unsafe content, while the unreasonable refuse rate indicates the proportion of instances where the model incorrectly refuses to answer questions that should be answered normally or positively guided. Only when both metrics are sufficiently low does it indicate that the model's overall safety is good, with minimal negative impact on its general capabilities. On our proprietary safety dataset of about 2000 test samples, our final INF-34-Chat reduces harmful proportion from 5.26% to 4.81% and unreasonable rejection rate from 6.24% to 4.35% after RL training compared with the SFT-only model.

3.1.3. Training Details

The stable dataset $\mathcal{D}$ we used for model training includes approximately 0.6M instructions, with its distribution outlined in Figure 8. Given the dynamic nature of data distribution and volume during the SFT across different application scenarios, our emphasis shifts from training epochs to specific training steps. The stable version of our model undergoes 1,200 training steps with a batch size of 32 and employs a cosine learning rate schedule ranging from 5e-5 to 1e-6. Additionally, we implement a 200-step learning rate warmup to ensure stability and a 0.1 dropout rate in hidden layers to mitigate overfitting.

Consistent with previous studies [48, 49], we perform the typical SFT in an autoregressive manner across all assistant response prompts y with total T tokens, without optimizing the loss for system and user prompts x:

$$\min_{\theta} -\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathcal{D}} \sum_{i=1}^{T} \log p_\theta(y_i|\mathrm{y}_{<i},\mathrm{x}). \tag{1}$$

To enhance training efficiency, we pack the prompts into 32K context windows, treating each packed instructions as a single data point and resetting attention mask exclusively for each instruction data separation. During experiments, we observed that integrating long context instruction data destabilized the SFT training due to imbalanced training tokens across different data points. To address this, we developed a balanced packing algorithm that equitably distributes the learnable tokens among data points and adjusted the loss aggregation from data points level to tokens level.

3.2. RLHF

We further align our LLM using Reinforcement Learning from Human Feedback (RLHF) following the Supervised Fine-Tuning (SFT) phase. RLHF entails fitting a reward model to a dataset reflecting human preferences and then optimizing the policy to elicit high-reward responses while minimizing deviation from the established SFT policy. In this section, we detail the technical specifics of our RLHF stage and highlight key differences from previous implementations.

3.2.1. Reward Model

Reward models, which guide policy improvement in the RLHF (Reinforcement Learning from Human Feedback) procedure [55], play a crucial role in model alignment. These models are instrumental in ensuring that the LLM's responses adhere to human ethical standards and preferences, covering aspects like safety, helpfulness, and mathematical reasoning. Unlike the common pairwise comparison of two model responses, as seen in RLHF methods like DPO [56] and PPO [57], we ask annotators to score each response individually on a scale from 0-10, following predefined guidelines. In instances where responses receive equivalent scores, annotators are further asked to determine which response aligns better with the task requirements.

Our reward model is composed of building a linear projection head on top of the last hidden state of the base model. The objective function includes the squared loss between the output value and the score, plus a pairwise loss that is applied only when the ground truth scores are tied. This approach involves directly regressing on the helpfulness or safety scores and relying on Bradley-Terry (BT) model assumptions to reconstruct the pointwise reward when the differences between two responses are nuanced. Reflecting on the trade-offs between safety and helpfulness as documented in previous studies [22, 58], we train separate models for safety and helpfulness to fine-tune our reward alignment strategy.

3.2.2. RLHF Training

Given the reward models, the subsequent step is to train our LLM with reinforcement learning algorithm. We leverage the standard off-policy REINFORCE [59] method rather than PPO or DPO-like methods. We update our parameter in the direction of :

$$\mathbb{E}_{q\sim q_{pool}}\mathbb{E}_{a\sim\pi_{ref}}\min\left(\frac{\pi_\theta}{\pi_{ref}},\rho\right)\nabla\log\pi_\theta(a|q)\left(\frac{r(a|q)-\bar{r}(q)}{r_{std}(q)}\right)-\lambda_1\nabla\mathbb{E}_{q\sim q_{pool}}KL(\pi_{ref}||\pi_\theta)-\lambda_2\nabla\mathbb{E}_{q\sim q_{sft}}KL(\pi_{sft}||\pi_\theta),$$

where π_θ is the current LLM that we aim to optimize, q is a question sampled from our question pool, and a is the corresponding answer sampled from the reference policy π_{ref}. The term $\frac{\pi_\theta}{\pi_{ref}}$ represents the importance sampling ratio to ensure the objective function is an unbiased estimator with respect to the corresponding on-policy objective function. The ratio is typically clipped to reduce variance, particularly when π_θ deviates significantly from π_{ref} and the sequence is long. Additionally, we include two KL divergence terms, i.e., $KL(\pi_{ref}||\pi_\theta)$ and $KL(\pi_{sft}||\pi_\theta)$, to ensure that the learned abilities do not degenerate during the RLHF stage. The reward $r(a|q)$ is obtained through either the reward model or verifiers such as ground truth label matching in GSM8K and unit tests in Python programs. To reduce the variance of the policy gradient, $\bar{r}(q)$, known as the baseline, is estimated by averaging the scores of the responses. Furthermore, the reward is normalized by $r_{std}(q)$, which is the standard deviation of responses' rewards given a question. Generally, we sample 10 responses for each question and use $\rho = 1.0$, $\lambda_1 = 0.2$, and $\lambda_2 = 1$ in our objective function. We perform several iterations of the

off-policy REINFORCE, transitioning the policy from π_0 (the SFT model) to π_1, π_2, and so on. In each iteration i, we set $\pi_{ref} = \pi_{i-1}$. This multi-iteration strategy is necessary to progressively enhance both the helpfulness and safety of the model.

3.3. Evaluation

In this section, we present the performance results of our chat model and other LLMS on various standard benchmarks, as well as two domain-specific benchmarks. We observe some discrepancies between our evaluation results of other open-source LLMs and the results reported in their respective technical reports. These discrepancies may be attributed to variations in the prompt formulation and post-processing procedures employed during evaluation.

3.3.1. Instruction-Following Benchmarks

We conducted evaluations using several automatic instruction-following benchmarks [60–63] and present the zero-shot results in Table 5, along with the zero-shot results of two mathematics tasks. These results reflect the chat model's ability to accurately follow instructions without prior examples. Our Inf-34B-Chat model is comparable to other open-source models of similar size.

Model	MT-bench	AlignBench	IFEval	Arena-Hard	GSM8K	MATH
Qwen1.5-32B-Chat	8.3	7.1	49.54	24.2	81.42	42.28
Yi1.5-34B-Chat	8.5	7.2	58.04	42.6	79.45	54.06
INF-34B-Chat	8.3	7.1	59.70	43.1	84.04	51.48

Table 5 | Zero-shot results of the chat models on instruction-following and mathematics benchmarks.

3.3.2. Industry Domain Benchmarks

We also evaluated our model's proficiency in the finance domain using the Chartered Financial Analyst (CFA) exam and in the medical domain using the United States Medical Licensing Examination (USMLE).

The USMLE is a rigorous, standardized examination that all physicians must pass to practice medicine in the United States. It comprises three steps, each with a specific focus and unique objectives, collectively ensuring a comprehensive assessment of a medical professional's competency. Step 1 evaluates the foundational medical knowledge of medical students; Step 2 assesses basic clinical knowledge; and Step 3 appraises advanced clinical knowledge and its application. Our model demonstrated impressive performance on the USMLE, as shown in Table 6: achieving scores of 79.83 on Step 1, 77.50 on Step 2, 81.75 on Step 3, and an overall score of 79.70. These results significantly surpass those of larger models, such as Qwen2-72B-chat, by a substantial margin.

CFA 2.0 is the Chartered Financial Analyst (CFA) exam dataset developed by the Shanghai Academy of Artificial Intelligence for Science (SAIS). It contains 200 hand-selected problems from each of the Level I and Level II CFA exams. Table 6 demonstrates that our model excels in financial analysis and surpasses other models.

3.3.3. Long Context Evaluation

Longbench LongBench[64] comprises 21 datasets spanning 6 task categories: Single-document Question Answering (Single-doc QA), Multi-document Question Answering (Multi-doc QA), Summarization, Few-shot Learning, Synthetic tasks, and Code Completion. These tasks collectively cover

Model	CFA 2.0	USMLE	Step 1	Step 2	Step 3
Qwen2-72B-Chat	62.00	70.53	68.07	70.83	72.26
Qwen1.5-32B-Chat	35.50	58.70	52.90	55.80	66.40
Yi1.5-34B-Chat	42.50	55.84	53.78	50.83	62.04
INF-34B-Chat	62.75	79.70	79.83	77.50	81.75

Table 6 | Results of the chat models on finance and healthcare benchmarks.

essential domains in long-text applications. The datasets are bilingual, featuring both Chinese and English languages, designed to assess models' capabilities in bilingual long-text processing. Specific results are provided in the Table 7. We used the same ID notation mentioned in LongBench for convenience and run evaluation tasks via OpenCompass [5]. We didn't test Yi1.5-34B as its context window length is 4K, and cited the results of GPT-3.5 from [64]. Our model has demonstrated superior performance on LongBench tasks compared to the equivalently scaled Qwen1.5-32B model [65].

NIAH(Needle in A Haystack) The Needle In A Haystack test [5, 66] is an evaluation method that randomly inserts key information into long texts to form prompts for large language models. The test aims to detect whether large models can extract such key information from extensive texts, thereby assessing the models' capabilities in processing and understanding long documents.

The visualization of evaluation results of INF-34B-Chat and Qwen1.5-32B-Chat are shown in Figure 9. Both INF-34B-Chat and Qwen1.5-32B-Chat performs well across context window lengths up to 32k on Single-Needle Retrieval Task [5].

4. Conclusion and Future Work

We released INF-34B, a family of large language models having 34B parameters, to the open source community, along with a permissive license to facilitate both research and commercial use. The INF-34B models yield very strong performance on widely-used benchmarks. We also released evaluation scripts to help reproduce the results. The INF-34B models with quantization support achieve a good balance between performance and cost and can serve as a solution for low-resource applications. Notably in two industry-domain tests CFA and USMEL, our model performs much better than two equally sized open access models and is even comparable to a model of twice the size, while our proprietary Trustworthy LLMs [1] achieve significant improvements further.

There are plenty of challenging explorations for future work. We continue to further enhance reasoning capacities via neural symbolic computing along the direction outlined in [1] for trustworthy LLMs, which introduces logical induction and deduction like System-II into the probabilistic predictions produced by LLMs, and also produce theoretically unlimited high-quality synthetic data for continuous model training. We delve into the structure of attention mechanisms to analyze the generation process, and couple with symbolic AI to improve explainability and transparency in content generation. We may explore alternative architectures for LLMs to address limitations in the attention mechanism. We also plan to release a manual on prompt engineering as a guidance to help users explore the potential of the released models.

Tasks/Model	ID	GPT-3.5-Turbo-16K	Qwen1.5-32B-Chat	INF-34B-Chat
Single-Doc QA	1-1	23.6	21.2	30.7
	1-2	43.3	43.7	44.4
	1-3	52.3	52.5	52.5
	1-4	61.2	65.0	62.1
	Avg	45.1	45.6	**47.4**
Multi-Doc QA	2-1	51.6	57.1	58.3
	2-2	37.7	43.2	49.7
	2-3	26.9	34.0	32.6
	2-4	28.7	27.3	32.1
	Avg	36.2	40.4	**43.2**
Summarization	3-1	29.5	31.8	30.7
	3-2	23.4	23.3	23.1
	3-3	26.7	22.8	25.7
	3-4	16.0	14.6	17.0
	Avg	23.9	23.1	**24.1**
Few-shot Learning	4-1	68.0	50.0	78.0
	4-2	91.4	88.9	92.3
	4-3	41.7	36.0	45.7
	4-4	29.2	35.5	48.0
	Avg	57.6	52.6	**66.0**
Synthetic	5-1	4.5	5.8	2.0
	5-2	71.0	98.0	98.5
	5-3	77.5	98.0	100.0
	Avg	51.0	**67.3**	66.8
Code	6-1	54.7	52.6	64.6
	6-2	53.6	35.0	49.8
	Avg	54.1	43.8	**57.2**
EN	Avg	44.0	44.0	**49.2**
ZH	Avg	44.5	47.4	**52.7**
All	Avg	44.7	45.5	**50.8**

Table 7 | Results (%) on single-doc QA, multi-doc QA and summarization tasks, few-shot learning, synthetic, and code tasks. 'Overall' is computed by the macro-average (the mean of 'Avg') over major task categories. This is computed on English (EN) tasks, Chinese (ZH) tasks, and all (All) tasks, code tasks are included in both languages.

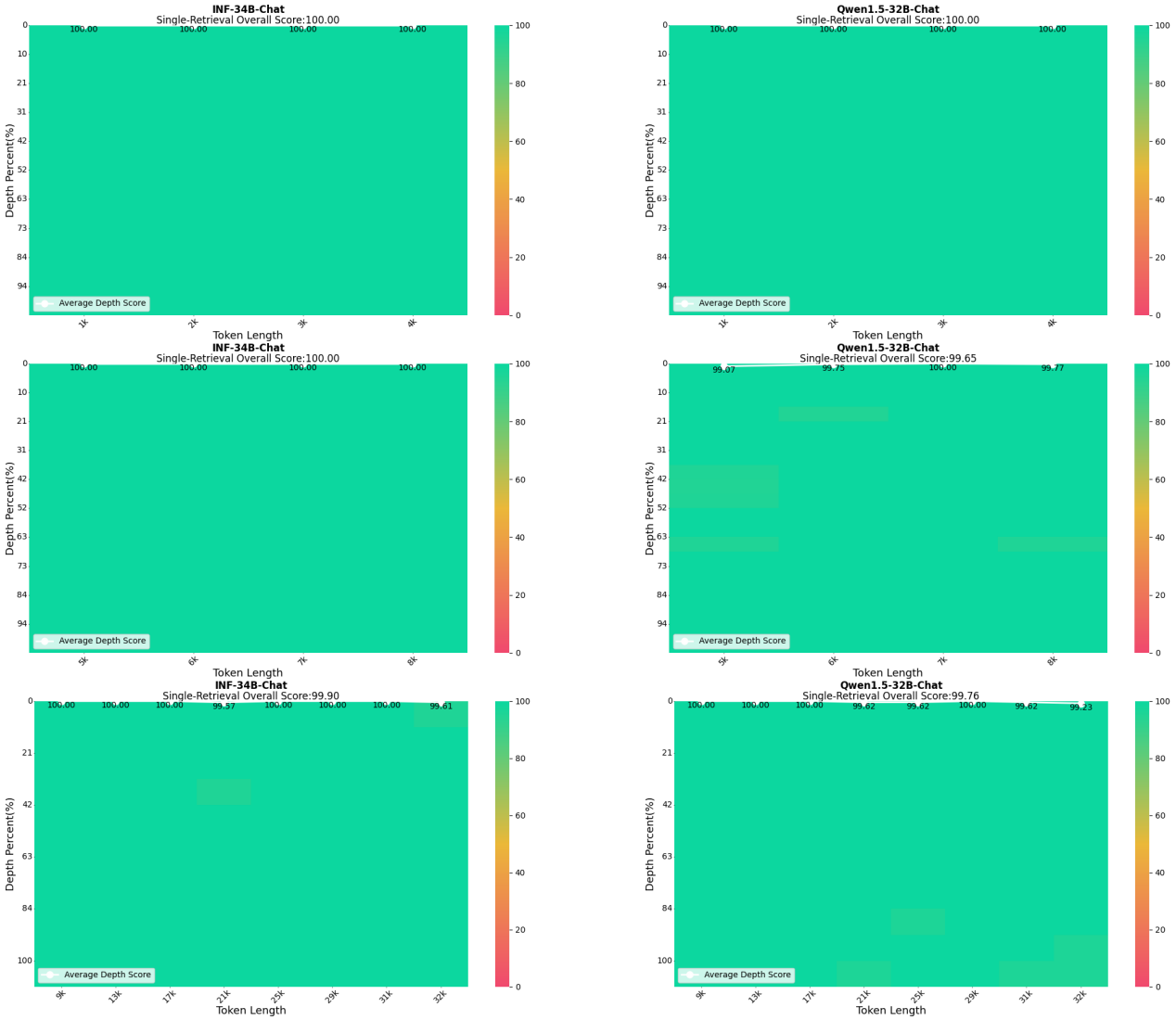

Figure 9 | Single-Needle Retrieval Task results of INF-34B-Chat and Qwen1.5-32B-Chat, where the green color indicates successful retrievals and the horizontal axis indicates token length varying from 0K to 32K.

A. Filters of General Data Pipeline

- **Document Sentence Count**: Documents containing only a single sentence were excluded.
- **Document Word Count**: Documents with fewer than 50 words were filtered.
- **Mean Word Length**: For English, we keep the documents with an average word length between 3 and 10. For Chinese, the range is from 1.3 to 10.
- **Nonconsecutive Special Characters**: Documents with more than 10% words containing non-consecutive hash marks (#) or ellipses (...) were removed.
- **Lines Ending with "read more"**: For English, documents where more than 10% of lines end with ellipsis words such as "..." and "read more" were excluded. For Chinese, the filter threshold changes to 30%, and with additional ellipsis words, e.g., "更多".
- **Bullet Points**: Documents where more than 90% lines start with bullet points were removed.
- **Punctuation**: Documents without punctuation marks were excluded.
- **Numerical Words**: Documents with more than 30% numeric words were filtered out.
- **Unigram Entropy**: Documents whose unigram entropy is less than 3 were excluded.
- **Duplicate Lines**: Documents with than 30% duplicated lines or 20% duplicated characters were removed.
- **Top N-grams**: Documents whose most frequent 2-gram (3-4 gram) exceeded 20% (18%, 16%) were filtered out.
- **Duplicated N-grams**: Documents where the most frequent 5-gram (6-10 gram) tokens occupy more than 15% (14-10%) were removed. For Chinese, the thresholds are 60%.
- **URL Blacklists**: For all documents with URLs, we filter the documents when their URLs are in the blacklist as introduced in [15].
- **URL Block Words**: We collected a list of NSFW block words, and remove the documents by these words as in [15].
- **Language Score**: We perform language identification using FastText [67] and remove English documents whose language score is below 0.65 and Chinese documents of 0.90.
- **Non-alphabetic Words**: Documents with over 50% of words not containing any English alphabetic characters were removed. Different from 20% in [15], we tolerate more non-alphabetic words to keep vast domain data. It is invalid for Chinese.
- **All Caps Words**: Documents with more than 50% of words in all caps were filtered out. It is invalid for Chinese.
- **Lorem Ipsum**: Documents with more than 2% of "lorem ipsum" characters were excluded. It is invalid for Chinese.
- **Stop Words**: Documents with less than 2 stop words from [68] are moved. It is invalid for Chinese.
- **Bracketed Words**: Documents with more than 10% words in full brackets (【】) were excluded.

B. Details of Domain Data Pipeline

B.1. Code-like Data

We utilize the GPT-4 model and employ the Autonomous Data Selection method to acquire relevant data from GitHub and the Common Crawl (CC). The Meta prompts used are shown below.

```
<system>You are ChatGPT, the most capable large language model equipped with extensive expertise in
mathematics and coding, particularly skilled in complex reasoning and problem-solving.
In the following interaction, I will provide you with a text script from a website.
Your task is to evaluate whether this script contains elements of code intelligence
 and if it is suitable for educational purposes for YOURSELF in the field of code.
Please respond with only YES or NO
<\system>
User: {
   "Text Script": "{text}"
}
1. Does the text contain elements of code intelligence? Reply with only YES or NO
2. Is the text suitable for educational purposes for YOURSELF in the field of code? Reply with only YES or
NO
Assistant: 1.
```

Figure 10 | The Meta Prompt for Selecting code-like seed from Github Markdown

```
<system>You are ChatGPT, the most capable large language model equipped with extensive expertise in
mathematics and coding, particularly skilled in complex reasoning and problem-solving.
In the following interaction, I will provide you with a text script from a website.
Your task is to evaluate whether this script contains elements of mathematical intelligence
 and if it is suitable for educational purposes for YOURSELF in the field of mathematics.
Please respond with only YES or NO
<\system>
User: {
   "Text Script": "{text}"
}
1. Does the text contain elements of mathematical intelligence? Reply with only YES or NO
2. Is the text suitable for educational purposes for YOURSELF in the field of mathematics? Reply with only
YES or NO
Assistant: 1.
```

Figure 11 | The Meta Prompt for Selecting math-like seed from Github Markdown and CC

The manual annotation of the URLs of the website is presented as shown in the table 8.

B.2. Math-like Data

The table 9 illustrates the distribution of Chinese code retrieved from Common Crawl in various domains.

Domain	Prefix	Tag
cloud.tencent.com	%cloud.tencent.com/developer/article%	Code
cloud.tencent.com	%cloud.tencent.com/ask%	Code
cloud.tencent.com	%cloud.tencent.com/developer/information%	Code
cloud.tencent.com	%cloud.tencent.com/document%	Code
my.oschina.net	%my.oschina.net%blog%	Code
ask.csdn.net	%ask.csdn.net/questions%	Code
www.cnblogs.com	%www.cnblogs.com%	Code
forum.ubuntu.org.cn	%forum.ubuntu.org.cn%	Code
q.cnblogs.com	%q.cnblogs.com/q%	Code
segmentfault.com	%segmentfault.com/q%	Code
segmentfault.com	%segmentfault.com/a%	Code
woshipm.com	%woshipm.com/data-analysis%	Code
zgserver.com	%zgserver.com/server%	Code
zgserver.com	%zgserver.com/linux%	Code
zgserver.com	%zgserver.com/ubuntu%	Code
juejin.cn	%juejin.cn/post%	Code
jiqizhixin.com	%jiqizhixin.com/articles%	Code
help.aliyun.com	%help.aliyun.com/zh%	Code
jyeoo.com	%jyeoo.com%	Math
www.haihongyuan.com	%haihongyuan.com%shuxue%	Math
www.03964.com	%www.03964.com%	Math
www.nbhkdz.com	%www.nbhkdz.com%	Math
9512.net	%9512.net%	Math
lanxicy.com	%lanxicy.com%	Math
bbs.emath.ac.cn	%bbs.emath.ac.cn%	Math
math.pro	%math.pro%	Math
mathschina.com	%mathschina.com%	Math
shuxue.chazidian.com	%shuxue.chazidian.com%	Math
shuxue.ht88.com	%shuxue.ht88.com%	Math

Table 8 | The manual annotation of code-like and math-like Chinese domains. We employ the '%' symbol as a wildcard in our Pattern matching. For instance, the URL 'https://my.oschina.net/u/4/blog/11' can be matched by the pattern '%my.oschina.net%blog%'.

B.3. Wiki-like data

Heuristic rules for wiki data cleaning Specifically, we remove all samples shorter than 128 characters, and use a keyword blacklist to remove web novels, company introductions, and biographies, while retaining knowledge-rich and educationally significant data, particularly in the fields of medicine, finance, and mathematics.

Details of fasttext model setting To accommodate fastText's reliance on spaces for tokenization, we use the byte pair encoding (BPE) tokenizer from INF-34B, which allows control over the vocabulary size. The specific model training settings include: the open-source fastText library, a vector dimension of 512, a learning rate of 0.1, a maximum n-gram length of 3, and a maximum word occurrence of 3.

Details of Constructing Fine-grained CC The Common Crawl (CC) dataset comprises an extensive collection of web pages. Traditionally, our approach to processing CC data has been limited to filtering and deduplication, which has hindered more advanced analysis of this vast dataset. To identify

reliable and high-quality data across various domains, a more sophisticated processing method is required. We propose a fine-grained division strategy for CC data. Initially, we segment the URLs in all snapshots of the CC dataset by base URL (e.g., www.google.com is considered a base URL). We count the occurrences of each base URL and rank them in descending order. Our findings indicate that the top 2 million base URLs account for approximately 65% of the entire CC dataset. We believe that annotating these base URLs with their type, topic, and language will provide valuable insights. Although this method allows for a preliminary fine-grained segmentation of the CC data, it may introduce some inaccuracies. Using this approach, we can extract most of the URLs related to Wikipedia from the CC dataset. Additionally, we can obtain substantial data for any domain. In the future, we will open source more details about this fine-grained CC data.

Wiki-like data annotation prompt To acquire educational content, we utilize a LLM API for annotation, similar to the Fineweb-edu [69] approach.

Domain	Code Recall Ratio (%)	Recall Count
blog.csdn.net	77.99	506457
cloud.tencent.com	44.03	148512
download.csdn.net	48.76	111634
www.csdn.net	76.85	106362
bbs.csdn.net	68.42	99497
my.oschina.net	73.41	80226
ask.csdn.net	92.03	69677
www.cnblogs.com	77.35	66153
blog.51cto.com	61.49	61845
support.microsoft.com	83.48	55295
www.oschina.net	61.22	54293
developer.aliyun.com	61.91	50812
zgserver.com	99.98	39592
forum.ubuntu.org.cn	42.21	37394
cn.depositphotos.com	95.69	35785
docs.amazonaws.cn	97.85	31155
yq.aliyun.com	49.05	30192
juejin.cn	78.21	29237
support.office.com	81.04	28386
zh.wikipedia.org	2.51	27996
experienceleague.adobe.com	83.62	27886
www.jianshu.com	12.72	25982
docs.vmware.com	85.29	25185
segmentfault.com	73.62	24877
www.infoq.cn	58.51	23804
wenjiangeshi.cn	99.92	21904
www.tripadvisor.com.tw	2.95	21828
www.03964.com	20.51	20793
blog.itpub.net	39.44	19841
linux.cn	53.08	19693
help.aliyun.com	74.44	19403
ithelp.ithome.com.tw	50.58	19293
androidcookie.com	99.63	18409
q.cnblogs.com	84.45	18319
aws.amazon.com	61.45	17703
wenku.csdn.net	73.78	17679
cn.tripadvisor.com	3.33	16790
www.eeworld.com.cn	6.95	16651
juejin.im	80.64	16261
lanxicy.com	12.86	15119
www.nbhkdz.com	48.10	14941
docs.microsoft.com	90.77	14676
www.jb51.net	57.95	14473

Table 9 | The distribution of the top websites in chinese code-like dataset

C. Details of Code Data Pipeline

C.1. Supported Programming Languages

ada, agda, alloy, antlr, applescript, assembly, augeas, awk, batchfile, bluespec, c, csharp, clojure, cmake, coffeescript, commonlisp, cpp, css, cuda, dart, dockerfile, elixir, elm, emacslisp, erlang, fsharp, fortran, gisl, go, groovy, haskell, html, idris, isabelle, java, javaserverpages, javascript, json, julia, kotlin, lean, literateagda, literatecoffeescript, literatehaskell, lua, makefile, maple, markdown, mathematica, matlab, ocaml, pascal, perl, php, powershell, prolog, protocolbuffer, python, r, racket, restructuredtext, rmarkdown, ruby, rust, sas, scala, scheme, shell, smalltalk, solidity, sparql, sql, stan, standardml, stata, systemverilog, tel, tcsh, tex, thrift, typescript, verilog, vhdl, visualbasic, xslt, yacc, yaml, zig

C.2. Workflows

The workflow of our code data pipeline is unveiled in Figure 12. We will elaborate each process in the following sections.

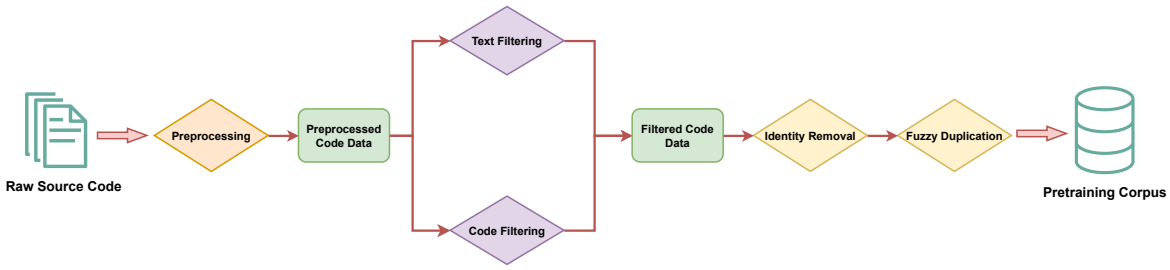

Figure 12 | The illustration of our code data processing workflow.

Preprocessing We exclude files exceeding 8 MB in size, as these are predominantly non-text files that incur considerable resource overhead. Furthermore, given the miscellaneous file types present on GitHub, we restrict our selection to files associated with 88 commonly used programming languages (App. C.1), by only retaining those files that have a corresponding file extension.

Heuristic Filtering The quality of the original code files on GitHub exhibits significant variability, with parts of lower quality code potentially affecting the LLM pre-training process. Referring to RefineWeb [15], We filter the data using heuristic filters to avoid introducing undesirable biases.

Files in our source code dataset are broadly categorized two main categories: text files and code files. For files in natural language, e.g., text and markdown files, we follow the strategy in Starcoder [70] and only keep the files whose filename contains "requirement" or matches "readme", "notes", "todo", "description", "cmakelists", to remove unrelated materials. Subsequently, we apply filtering rules similar to those used in general datasets, with minor adjustments to the quotas to accommodate the specific needs of our code dataset.

For the code data, we use a set of specific code filters, since the characteristic of code files is substantially different from the general texts, making general filtering rules incompatible. Besides, the target abilities of LLM learning from general data and code data diverge significantly. Specifically, the desired outcomes from code data predominantly focus on the model's comprehension of code logic,

a contrast to the broader capabilities targeted by general LLM training. Founded on the principle applied to phi-1 [71], we have developed our filtering criteria based on the following insights: 1) Filter out files with poor self-containment; 2) Filter out files with little or poor code logic; 3) Filter out files with non-standard formatting. Based on these foundational insights, we establish three primary categories of rules that are delineated as follows.

Natural Language Filtering Rules: Natural language filtering rules inherit a part of the filtering rules utilised in general data processing pipeline, which overlap with those filters displayed in section 2.1.1 to some extent, such as removing files with excessively repetitive text and files with low letter proportions. This category of rules addresses the fundamental text quality criteria that are applicable to both general textual content and code, ensuring a baseline standard of data integrity across different file types.

Code Filtering Rules: Code filtering rules are specifically designed to address the common characteristics of various programming languages. Drawing upon filtering rules introduced in StarCoder V2 [30], we have developed a new suite of rules with more comprehensive code analysis, such as filtering out files with a high proportion of long strings or excessive number of hexadecimal constants.

Code-specific Filtering Rules: Code-specific filtering rules are devised based on the different programming languages. We have developed specific filtering rules tailored to these eight selected programming languages: Python, C, C++, C#, Java, JavaScript, Go, and HTML, which are commonly used or have high data proportions. For instance, we exclude Python files that contain an excessive proportion of "import" statements and C files that excessively use "goto" statements, aiming to eliminate low-quality code data specific to these predominant programming languages.

Deduplication The purpose of deduplication is to construct an unbiased and diverse training set while significantly reducing the data volume. We adopt an aggressive file-level deduplication strategy, since we found file-level deduplication outperforms repo-level version significantly. Owing to the extremely high repetition of the code dataset, we leverage both identity removal and fuzzy deduplication methods to eliminate documents containing identical or near-identical code content shown as follows:

Identity Removal: Due to the prevalence of forking and copying within the codebase, almost 90% files are completely duplicated. On account of this, identity removal is applied towards code data at the first step in this module. We compute the SHA256 hash value for each document and keep those with the highest star count and the latest commit time when the hash values collide.

Fuzzy Deduplication: Following the fuzzy deduplication setting in general data pipeline, we split raw text into 5-gram pieces, and then calculate the 2048 minhash functions [11]. Additionally, we utilize LSH [12] to retain only those distinct files with highest stars and latest commit time. This process removes 6% file volume.

C.3. Quality Verification

To demonstrate the efficacy of our code pipeline, we train a 1.5b code LLM up to 550B tokens using only our processed code data compared with using the source code data from the stack v2 [30]. The comparison of the training loss between the two models, as well as the evaluation metrics (HumanEval Pass@1 and MBPP Pass@1) during the training process, are presented in the Figure 13, Figure 14 and Figure 15. From these figures we find that LLM pretraining on our processed code has a significantly training loss and higher evaluation metrics, indicating that our code dataset outperforms the stack v2 in both diversity and quality.

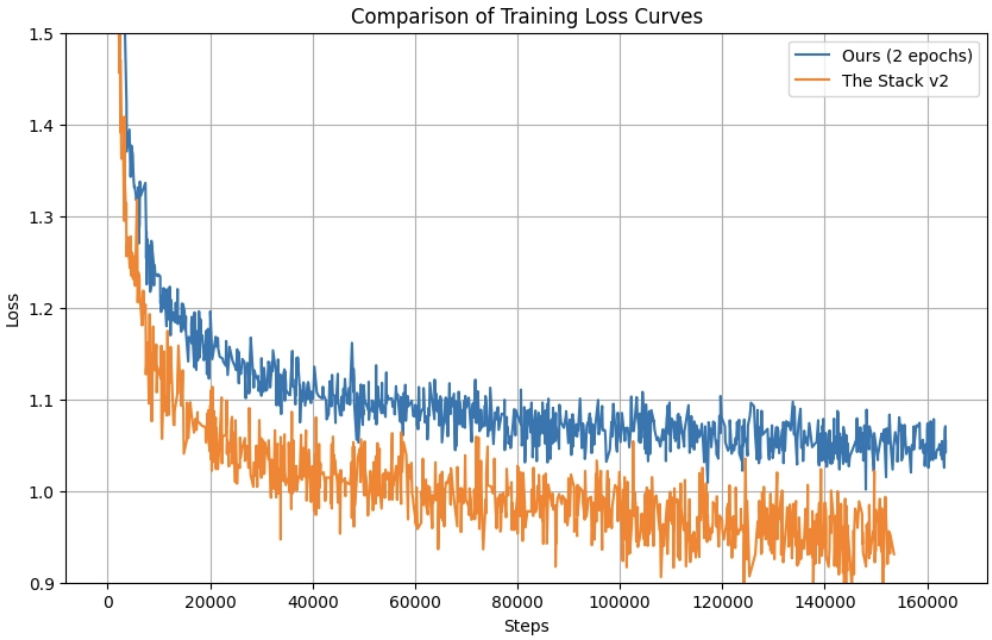

Figure 13 | The comparison of the training loss between using our code dataset and the Stack v2.

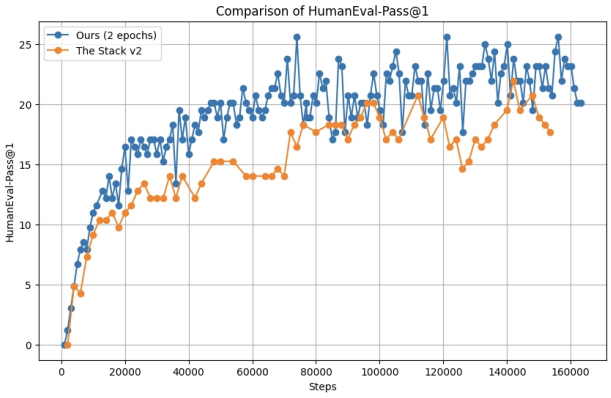

Figure 14 | The comparison of HumanEval Pass@1 between using our code dataset and the Stack v2.

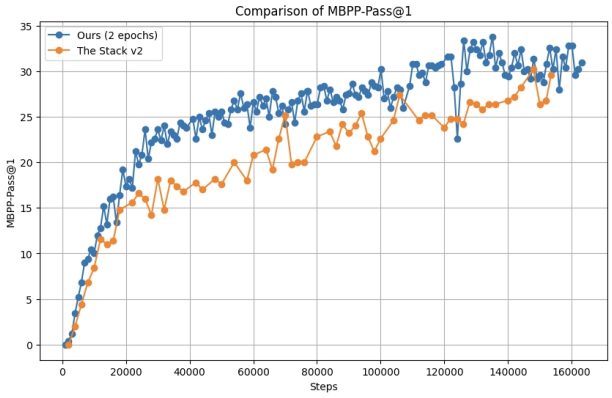

Figure 15 | The comparison of MBPP Pass@1 between using our code dataset and the Stack v2.

In the ongoing development of the code data processing pipeline, we aim to create a more comprehensive and robust iteration in the near future. This enhanced version will encompass a broader spectrum of code data and feature more refined data processing techniques. Due to current time constraints, the entire pipeline code, along with the processed, high-quality code data, will be made publicly available in our forthcoming work, which will specifically address advancements in code LLM. We expect that these contributions will significantly benefit the open-source code LLM community.

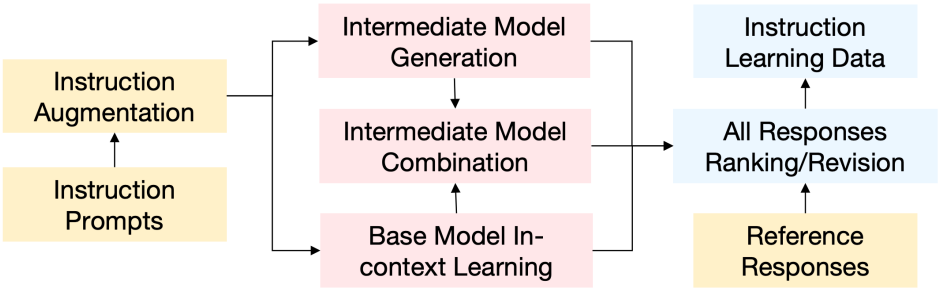

Figure 16 | The basic instruction prompts generation pipeline used for all prompts generation. Starting with the collection of initial prompts, we enhance these through instruction augmentation. We then apply intermediate models and ICL techniques to generate outputs. These outputs are combined by the intermediate model to create new candidate responses. Annotators then rank all candidate responses, including reference ones, and make necessary revisions to finalize the instruction data.

D. Prompt Preparation

Multiple open source datasets are available for LLM instruction fine-tuning [72–74]. However, merely aggregating these datasets can lead to significant variations in quality and quantity in different instructional tasks. This variation often results in unbalanced behaviors, including an increased incidence of hallucinations in model responses. To address this, our approach in prompt preparation emphasizes the creation of high-quality and diverse prompts and aims to enhance the generalization capabilities of LLM across a variety of tasks.

Measurement of Data Quality: Improving the data quality of instruction data is one of the most effective and imperative way to improve the SFT performance for both general chat capability and domain-specific tasks. However, key challenge lies in accurately measuring and assessing the quality of instruction data, which is essential for subsequent effective prompt generation and verification. To address this, we propose two indicators to measure the quality of prompts. The first indicator assesses textual quality, identifying issues such as truncation, formatting errors, language inconsistency, irrelevant content, unsafe material, non-compliance with user instructions, and unnecessary repetition. The second indicator evaluates the helpfulness of responses, focusing on their authenticity and accuracy. It ensures that responses are in the standard assistant form, and are free from misleading or contradictory information, incorrect assumptions, or biases about the user. Furthermore, it also prefer the response prompts that are not exhibit human-like characteristics such as personality, emotions, or preferences, nor claim real-life actions when not role-playing. Additionally, it also evaluate the provided code text to ensure that all contents are correctly formatted using markdown.

Measurement of Data Diversity: The diversity of instruction data is crucial for enhancing a model's ability to generalize across various tasks and follow instructions effectively. Our approach focuses on two key aspects of diversity: task distribution and data distribution. Task distribution refers to the segmentation of instruction tasks. To analyze and enhance task distribution diversity, we have developed a dual categorization system. This system includes a three-tiered topic categorization and a separate instruction categorization to organize the instruction data systematically and monitor task distribution trends. On the other hand, data distribution pertains to the textual characteristics of the instructions themselves, encompassing both the format and stylistic elements of the text. Our findings suggest that a thoughtfully designed task distribution can effectively balance between generality and specialization. Moreover, a diverse data distribution significantly reduces hallucinations and promotes the learning of invariant semantic features [75, 76] in SFT.

Initial Prompts Generation: Following the LIMA [50], we fist construct around 3K preliminary high quality instruction data based on the LIMA data distribution completely annotated and examined by all authors. Then, we perform SFT on this dataset and acquire the intermediate model for initial prompt generation. We initiate the prompt development process by collecting single-turn prompts from open source projects like Dolly [77] and COIG [78]. These prompts are categorized into nine types: creative writing, cloze QA, open QA, summarization, information extraction, classification, brainstorming, coding, math & reasoning, and few-shot instructions. Given the varying quality of data sources, we've designed a two-stage process to enhance prompt quality. In the first stage, we employ both intermediate model and the base model with in-context learning to create initial prompts [79–82]. Subsequently, we use chat-based intermediate model to refine these prompts by integrating and improving upon provided reference answers. In the second stage, we compile all refined responses and engage a team of human annotators to rank these responses and select the highest quality ones according to established data quality standards. Through several annotation experiments, we've identified two critical factors for improving annotation quality: allowing each annotator to specialize in a single category of instruction data, and prioritizing data selection and ranking over direct annotation. This approach not only enhances efficiency but also yields better results. Employing this comprehensive prompt generation process, we have successfully produced approximately 100K high-quality, single-turn general instruction prompts. By using these part of instruction data, we perform the initial SFT on our base checkpoint and utilize this model as the initial model for subsequent instruction data production.

System Prompts Generation: In order to enhance our model's capacity to perform diverse tasks and role-play in accordance with system prompts, we independently generated 15k of instruction data with system prompts. Compared with the instruction prompts we introduced in the previous part, system prompt can be considered as a universal instruction that can influence the entire dialogue. Specifically, to acquire a range of high quality and diverse system prompts, we filtered web pages using keywords and extracted prompts from the Common Crawl[6] data using chat-based intermediate model. Most of these system prompts were real prompts shared by users on social media platforms for their agents, spanning categories like role-playing and text adventure games. We initially gathered approximately 1.5k system prompts. To further diversify our model's adherence to these prompts, we perform data augmentation to enrich the formats which are commonly used by humans in crafting system prompts, including second-person narratives (e.g., "You are a..., your task is..."), Markdown, and YAML. Moreover, since the prompts sourced from the Common Crawl were predominantly in English, we than translated all non-Chinese prompts into Chinese and invite our annotation group to perform data selection following the aforementioned data filtering standards, resulting in approximately 15K instruction data with system prompts in various formats and languages.

Dialogue Generation: To further improve the LLMs dialogue and system prompt capabilities in controlling global dialogue information, we independently construct our dialogue instruction data within and without system prompts. We engaged trained annotators who initiated dialogues based on the system prompts and provided first instruction prompts, asking specific questions about the details within these prompts. Similar to the single turn instruction prompt generation, we sample the responses from intermediate models and our previously SFT checkpoint. After that, human annotators then rank the response, refine these responses to ensure higher quality, and repeat this process to continue with multi-turn dialogues. Due to the high work load of dialogue generation by annotators, we also incorporated synthetic data generation process [83, 84], employing an independent instance of intermediate model as the users, who engaged in multi-turn dialogues according to the system prompts, simulating real human interactions. We ultimately compiled about 20k multi-turn dialogues with half of them have system prompts.

[6] https://commoncrawl.org/

Instruction Augmentation: In our previous work, we developed a method to generate a variety of prompts for SFT. Despite these advancements, the model struggles with following complex instructions. To enhance its ability to understand and generalize instructions, we introduced an instruction augmentation process aimed at creating more complex prompts. This process begins by categorizing instructions into nine distinct types, without considering the topic information in the text: information retrieval, creativity, comparison/contrast, causality/relationship, logical reasoning, conditional, semantic understanding, restrictive, and sequential compound instructions. It is common for a single data point to encompass multiple instruction types. Unlike other research that employs a universal approach to instruction augmentation using identical prompts, our method tailors the augmentation to the specific type of instruction task with intermediate models. However, we observed that the quality of instructions generated synthetically was inferior to those crafted by humans. Consequently, we engaged our annotation team to refine these instructions. Following these revisions, we proceed with prompt production using our aforementioned data pipeline.

Data Preparation for Agentization: In supporting domain-specific applications, we discovered that agentizing LLMs is crucial for achieving accurate responses in complex practical scenarios. In the SFT stage, agentization involves preparing intention identification and tool utilization prompts. We developed four tool types for this purpose: search engines, text-to-image generation, image-to-text generation, and self-cognition, accessible via API. Each tool's input and output are managed in text, using special tokens to demarcate the start and end of API interactions, mimicking a textual sequence. This method, inspired by Toolformer [85], involves three steps: initiating the API call with a special token and relevant parameters, receiving the API's textual response, and generating the final model response based on this data. To further improve the intention recognition capability through SFT, we also prepare specific prompts through data backflow from applications and totally produce around 10K entries through our data pipeline.

E. ChatML Template

Similar to the approaches in related works [49, 86–89], we developed the Chat Markdown Language (ChatML) template to distinctly categorize system prompts, user prompts, and assistant prompts. This templating approach ensures uniformity across all subsequent instruction learning and inference scenarios, promoting consistent interaction behavior. The details of ChatML are presented in Table 10.

ChatML template for Infml's instruction learning
<|start|>system
{*system_prompt*}<|end|>
<|start|>user
{*user_prompt*}<|end|>
<|start|>assistant<|message|>{*assistant_prompts*}<|end|>
<|start|>user
{*user_prompt*}<|end|>
<|start|>assistant<|message|>{*assistant_prompts*}<|end|>

Table 10 | ChatML template for two turns dialogue with system prompts. The {*system_prompt*}, {*user_prompt*}, and {*assistant_prompts*} represent the placeholders for system prompts, user prompts, and assistant prompts.

In cases where the instructional data lacks system prompts, we utilize only the user and assistant components of the ChatML template. During inference, users have the option to either incorporate

system prompts using the default setting: "You are a helpful assistant." or to simply merge the user and assistant segments for straightforward instructional interactions.

F. Long Context Data Sources and Tasks

F.1. Data Sources and Processing

Arxiv Arxiv is an academic preprint platform that collects papers from fields such as mathematics, physics, and computer science, covering a wide range of topics. Texts on Arxiv typically adhere to specific structural norms, making it crucial for large language models to learn to extract information and understand the semantics embedded within these structures. We downloaded recent papers along with their corresponding LaTeX source files from Arxiv. Our first step involved extracting the core content of the articles from Latex source files. Subsequently, we conducted hierarchical extraction of the main content based on specific keywords (e.g., abstract, section, chapter). This process enabled us to organize the main body of text into structured formation. Following this, we filtered out papers where the main content was less than 512 tokens or did not include the methodology part after hierarchical extraction. Finally, the extracted content was then saved in JSON format for further processing and analysis.

Science Fiction Science fiction novels typically involve futuristic technologies, extraterrestrial life forms, supernatural phenomena, and other elements requiring high levels of imagination and complex worldviews. Models can enhance their reasoning abilities by understanding the internal logic of these narratives, while the complexity of the plots can improve the models' capacity for multi-layered information processing. We downloaded a collection of science fiction novels from 4675-scifi. Initially, we utilized regular expressions to remove watermarks interspersed within the novel texts. Upon inspecting the novel contents, we identified redundant information unrelated to the core narrative at the end of some original novels, which we manually deleted. Subsequently, we employed regular expressions and specific chapter keywords to segment the novel contents into chapters. After segmentation, we filtered out novels lacking complete chapter content to ensure the integrity of the final novel contents. Finally, we saved the segmented content in JSON format, organized by chapters.

Reddit Dialogue The Reddit platform hosts a wealth of user-generated content including dialogues, comments, and responses, which are commonly employed in natural language processing tasks such as dialogue generation, sentiment analysis, and topic classification. We extracted a significant volume of raw Reddit data from online sources, specifically targeting dialogues that exceed 128 tokens in length. This dataset serves as the foundational corpus for generating multi-turn dialogues in subsequent phases of our research.

Financial Reports and Comments Financial reports and their corresponding commentary data cover multiple industries and companies, as well as various financial and business metrics. This diversity and complexity contribute to enhancing the generalization capability and adaptability of models trained on such data across different contexts. The intricate syntactic structures and deep contextual dependencies present in these datasets enable models trained on them to handle complex sentence structures, multiple modifiers, and long-distance dependencies, thereby improving the models' ability to understand and generate lengthy texts. We downloaded a large volume of financial data and corresponding financial report commentaries from the CNINFO website. Recognizing potential discrepancies between commentaries and financial reports, we initially employed intermediate version of INF-34B-32K model trained on open source long context datasets to remove portions from the commentary that are not present in the financial reports. Subsequently, we conducted manual checks to ensure that the remaining commentary information is consistent with the content found in the financial reports.

Open Sources Long Context Dataset We also integrated several open-source long-text datasets such as LongAlpaca[90], LongAlign[91], anti-haystack[92], etc. For these datasets, we initially selected appropriate segments based on their length and conducted random sampling checks to identify redundant characters that could potentially degrade text quality. Subsequently, we employed regular expressions to systematically traverse the content and eliminate any adverse effects identified during the sampling process.

F.2. Task Diversity

Summarization Based on financial reports and commentary data, we constructed a dataset akin to an summarization task. The query component involves specific content from financial reports, prompting an analytical critique of the report. The answer section is composed of information that has undergone manual quality assurance and is fully traceable within the financial reports.

Multi-Turn Dialogues For English dialogues, we utilize preprocessed Reddit data as described above and employ INF-34B to select conversation content involving two individuals. These conversations are used respectively as references for the dialogue between a User and an AI Agent, constituting a dialogue exchange between them. The User's dialogue strives to adhere closely to the original conversation content, introducing additional information outside the original context if necessary to clarify ambiguous references and prevent misinterpretations. The AI Agent's dialogue may selectively draw from the original content while maintaining characteristic traits such as helpfulness, honesty, harmlessness, and truthfulness.

For Chinese dialogues, we directly generate using INF-34B. Initially, different roles are assigned to the model, representing entities with knowledge or experience in various domains. Suitable prompts are used to generate the first line of dialogue for both the User and AI Agent, termed as the "hook". Finally, based on the hook and the background of each role, appropriate templates are applied to generate a complete dialogue between the User and the AI Agent (in its respective roles).

Multiple Information Retrieval Based on financial reports sourced from the CNINFO website, we utilized intermediate version of INF-34B-32K to construct datasets tailored for extraction tasks. These tasks encompassed the extraction of specified financial data types from the content of financial reports, extraction of the narrative structure of entire financial reports, and tasks aimed at extracting content for financial trend analysis from the reports. Following the construction of these datasets, a rigorous manual verification process was conducted to ensure that the extracted content accurately corresponds to the information present in the original financial reports. The subsequent manual inspection phase served to validate that the extracted data aligned faithfully with the corresponding sections of the source financial reports, thereby affirming the reliability and fidelity of the constructed dataset for subsequent analytical and model training purposes.

Question-Answer We utilized preprocessed contents of ArXiv articles to construct various types of question-answer tasks. These included numerical response questions, opinion-based questions requiring judgments based on the content, and multiple-choice questions based on content(potentially containing multiple correct options). For questions where the correct answer is numerical, we employed intermediate version of INF-34B-32K to generate corresponding questions and answers based on the original content. Subsequently, manual verification was conducted by searching for the answers within the original content to ensure accuracy.

For opinion-based questions derived from content, intermediate version of INF-34B-32K generated both correct and incorrect conjectures about the given content. During the data construction phase, a random selection was made between these conjectures for each Arxiv article.

Regarding multiple-choice questions, we extracted factual information from the abstract sections of corresponding articles. Using both the extracted abstract facts and the main body of the articles, we utilized intermediate version of INF-34B-32K generated multiple-choice questions.

Comprehension and Reasoning We filtered out books from the preprocessed ArXiv dataset and merged them with the collection of science fiction novels. Using intermediate version of INF-34B-32K, we generated question-answer pairs based on multiple chapters, requiring that correct answers derive from understanding across at least two chapters rather than relying solely on a single chapter. By focusing on questions where correct answers necessitate insights synthesized from multiple chapters, we aim to develop datasets that reflect nuanced comprehension and contextual integration in textual analysis tasks.

G. Quantization Results

GPTQ and AWQ are Post-Training Quantization (PTQ) methods that save memory and provide potential speedups while retaining the model's accuracy. To run GPTQ or AWQ, we use [AutoGPTQ](https://github.com/AutoGPTQ/AutoGPTQ) and [AutoAWQ](https://github.com/casper-hansen/AutoAWQ). Additionally, the Hugging Face transformers library has integrated Optimum to perform quantization on language models. We provide quantized models in three formats: GPTQ4bit, GPTQ8bit, and AWQ4bit.

Performance Benchmark

Table 11 presents performance of our bf16 model and quantized models (including GPTQ-Int4, GPTQ-Int8, and AWQ-Int4). Specifically, we report the inference speed (tokens/s) and memory usage (GB) under different context length conditions. All results are measured on a web service launched on a single GPU using vLLM.

Quantization	Parallel	Total Requests	Total Time (s)	Total Prompt	Total Output	Average Prompt	Average Output	Avg ttft (ms)	Avg Tokens/sec
BF16	1	8	55.24	7656	1116	957.0	139.5	294.19	21.16
BF16	2	16	55.20	15312	2055	957.0	128.44	346.44	19.85
BF16	4	32	65.36	30624	4242	957.0	132.56	392.19	17.77
BF16	8	64	78.51	61248	8416	957.0	131.5	482.47	14.82
BF16	16	128	102.41	122496	16718	957.0	130.61	650.55	11.13
BF16	32	256	179.99	244992	33189	957.0	129.64	7392.42	9.11
AWQ 4bits	1	8	36.56	7656	1116	957.00	139.50	469.96	34.16
AWQ 4bits	2	16	39.12	15312	2055	957.00	128.44	515.20	29.76
AWQ 4bits	4	32	50.93	30624	4242	957.00	132.56	574.12	23.73
AWQ 4bits	8	64	70.06	61248	8416	957.00	131.50	695.86	16.95
AWQ 4bits	16	128	106.45	122496	16718	957.00	130.61	925.55	10.79
AWQ 4bits	32	256	189.00	244992	33189	957.00	129.64	1394.61	5.94
GPTQ 4bits	1	8	23.95	7656	1116	957.00	139.50	339.06	52.89
GPTQ 4bits	2	16	28.60	15312	2055	957.00	128.44	370.04	40.63
GPTQ 4bits	4	32	39.99	30624	4242	957.00	132.56	420.13	29.96
GPTQ 4bits	8	64	65.86	61248	8416	957.00	131.50	531.12	17.63
GPTQ 4bits	16	128	119.64	122496	16718	957.00	130.61	739.55	9.38
GPTQ 4bits	32	256	226.12	244992	33189	957.00	129.64	1151.64	4.87
GPTQ 8bits	1	8	34.48	7656	1116	957.00	139.50	343.56	35.31
GPTQ 8bits	2	16	36.10	15312	2055	957.00	128.44	384.82	31.55
GPTQ 8bits	4	32	53.00	30624	4242	957.00	132.56	438.40	22.14
GPTQ 8bits	8	64	86.73	61248	8416	957.00	131.50	551.20	13.21
GPTQ 8bits	16	128	159.62	122496	16718	957.00	130.61	776.12	6.96
GPTQ 8bits	32	256	234.27	244992	33189	957.00	129.64	1163.31	4.73

Table 11 | Quantization Performance Metrics

Generation Results of Quantized Models

Table 12 reports the generation results of quantized models, including both GPTQ and AWQ. All models were evaluated using greedy decoding.

Testing Environment

Quantization	MMLU	CMMLU	C-Eval	ARC-c	GSM8K
BF16	76.45	81.26	82.29	90.17	84.08
GPTQ-Int4	75.16	79.57	79.47	89.49	82.49
GPTQ-Int8	76.63	81.28	82.23	91.19	84.61
AWQ-Int4	75.59	80.15	80.90	90.51	83.78

Table 12 | Generation Results

- NVIDIA A100 80GB
- CUDA 12.1
- Pytorch 2.3.0+cu121
- Flash Attention 2.5.0
- Transformers 4.42.4
- AutoGPTQ 0.8.0
- AutoAWQ 0.2.5
- vLLM 0.3.3

H. OpenCompass Evaluation

OpenCompass [5] is a flexible and efficient evaluation framework that users can easily install and utilize. This tool provides a comprehensive environment for assessing the performance of large models across various tasks. In our study, we employed OpenCompass to evaluate several common Natural Language Processing tasks. The detailed test results [7] are presented in Table 13 and Table 14. This evaluation serves as a reference point for comparing model performance.

[7] Please visit https://github.com/infly-ai/INF-LLM for more details.

	Qwen1.5-32B-Base	Yi1.5-34B-Base	INF-34B-Base
	Exam		
C-Eval	82.91	83.06	80.75
AGIEval	59.28	56.24	55.19
MMLU	73.78	77.70	76.11
CMMLU	82.43	83.67	80.08
GAOKAO-Bench	84.26	63.68	60.89
ARC-c	90.17	94.58	90.17
ARC-e	94.89	97.35	96.30
	Knowledge		
BoolQ	88.50	87.80	87.06
CommonSenseQA	73.46	71.74	70.93
TriviaQA	65.81	68.60	68.93
NaturalQuestions	8.56	14.99	37.81
	Understanding		
C3	96.38	92.11	94.14
RACE(Middle)	93.59	94.64	92.13
RACE(High)	89.79	85.45	87.59
OpenbookQA	93.60	89.40	93.80
CSL	62.50	61.88	66.25
LCSTS	17.99	16.77	17.48
XSum	21.46	42.46	41.31
EPRSTMT	91.25	88.75	91.25
LAMBADA	73.80	72.02	75.63
	Reasoning		
CMNLI	61.15	52.59	58.03
OCNLI	59.02	49.56	54.24
AX-b	57.52	58.61	43.75
AX-g	85.39	82.58	75.84
RTE	55.23	74.01	66.79
COPA	98.00	97.00	99.00
ReCoRD	31.61	70.96	62.89
HellaSwag	81.90	81.39	83.32
PIQA	81.72	81.88	81.61
SIQA	76.41	79.02	77.74
MATH	38.40	33.90	38.84
GSM8K	73.54	79.45	83.02
BBH	70.50	75.15	71.20

Table 13 | Evaluation results of base models using OpenCompass.

	Qwen1.5-32B-Chat	Yi1.5-34B-Chat	INF-34B-Chat
		Exam	
C-Eval	81.21	81.65	82.29
AGIEval	58.45	60.04	61.10
MMLU	75.09	75.93	76.45
CMMLU	80.17	80.40	81.26
GAOKAO-Bench	86.43	71.93	74.88
ARC-c	82.37	92.88	90.17
ARC-e	90.30	95.94	96.47
		Language	
WiC	68.50	64.26	67.87
CHID	89.11	81.68	83.66
AFQMC	73.24	73.08	72.24
WSC	80.77	70.19	68.27
TyDiQA	40.00	20.31	37.82
		Knowledge	
BoolQ	88.69	90.64	89.51
CommonSenseQA	87.55	86.49	85.50
TriviaQA	64.20	67.84	71.53
NaturalQuestions	33.91	33.57	37.09
		Understanding	
C3	95.89	95.01	94.36
RACE(Middle)	93.59	91.71	92.83
RACE(High)	91.31	88.82	89.37
OpenbookQA	92.60	94.20	95.00
CSL	53.75	53.75	50.00
LCSTS	18.92	12.91	15.30
XSum	29.54	21.16	23.66
EPRSTMT	91.88	91.25	90.62
LAMBADA	62.08	67.30	68.31
		Reasoning	
CMNLI	62.76	63.64	62.11
OCNLI	60.54	60.14	59.19
AX-b	68.57	75.09	60.33
AX-g	91.57	92.70	85.39
RTE	79.06	85.92	75.45
COPA	100.0	100.0	100.0
ReCoRD	44.77	9.22	65.35
HellaSwag	86.67	85.98	88.98
PIQA	86.45	88.14	84.39
SIQA	69.14	60.54	77.23
MATH	42.28	54.06	51.48
GSM8K	81.43	79.45	84.08
BBH	70.81	74.80	72.67

Table 14 | Evaluation results of chat models using OpenCompass.

References

[1] Inf Team. Towards trustworthy large language models in industry domains. Technical report, Inf, 2024. URL https://s.infly.cn/f/img/pdf/Towards_Trustworthy_LLMs.pdf.

[2] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. 2020. URL https://arxiv.org/abs/2001.08361.

[3] Meta. Llama 3. Technical report, Meta, 2024. URL https://ai.meta.com/blog/meta-llama-3/.

[4] Nemotron-4 340b technical report. Technical report, NVIDIA, 2024. URL https://research.nvidia.com/publication/2024-06_nemotron-4-340b.

[5] OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

[6] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023. doi: 10.48550/ARXIV.2308.12950. URL https://doi.org/10.48550/arXiv.2308.12950.

[7] Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Lin, Emmanouil Benetos, Huan Yang, Junting Zhou, Kaijing Ma, Minghao Liu, Morry Niu, Noah Wang, Quehry Que, Ruibo Liu, Sine Liu, Shawn Guo, Soren Gao, Wangchunshu Zhou, Xinyue Zhang, Yizhi Zhou, Yubo Wang, Yuelin Bai, Yuhan Zhang, Yuxiang Zhang, Zenith Wang, Zhenzhu Yang, Zijian Zhao, Jiajun Zhang, Wanli Ouyang, Wenhao Huang, and Wenhu Chen. Map-neo: Highly capable and transparent bilingual large language model series. *arXiv preprint arXiv: 2405.19327*, 2024.

[8] Xinrun Du, Zhouliang Yu, Songyang Gao, Ding Pan, Yuyang Cheng, Ziyang Ma, Ruibin Yuan, Xingwei Qu, Jiaheng Liu, Tianyu Zheng, Xinchen Luo, Guorui Zhou, Wenhu Chen, and Ge Zhang. Chinese tiny llm: Pretraining a chinese-centric large language model, 2024. URL https://arxiv.org/abs/2404.04167.

[9] Together Computer. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

[10] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *CoRR*, abs/2107.06499, 2021. URL https://arxiv.org/abs/2107.06499.

[11] Andrei Z. Broder. On the resemblance and containment of documents. In Bruno Carpentieri, Alfredo De Santis, Ugo Vaccaro, and James A. Storer, editors, *Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings*, pages 21–29. IEEE, 1997. doi: 10.1109/SEQUEN.1997.666900. URL https://doi.org/10.1109/SEQUEN.1997.666900.

[12] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets, 2nd Ed*. Cambridge University Press, 2014. ISBN 978-1107077232. URL http://www.mmds.org/.

[13] Udi Manber and Eugene W. Myers. Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993. doi: 10.1137/0222058. URL https://doi.org/10.1137/0222058.

[14] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2022.

[15] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data only. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/fa3ed726cc5073b9c31e3e49a807789c-Abstract-Datasets_and_Benchmarks.html.

[16] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. Wu Y.K. Li, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[17] Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web, 2024. URL https://arxiv.org/abs/2405.03548.

[18] Yifan Zhang, Yifan Luo, Yang Yuan, and Andrew Chi-Chih Yao. Automathtext: Autonomous data selection with language models for mathematical texts. *arXiv preprint arXiv:2402.07625*, 2024.

[19] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[21] Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *arxiv*, 2024. URL https://arxiv.org/abs/2407.13623.

[22] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[23] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

[24] 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. URL https://arxiv.org/abs/2404.06395.

[25] Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*, 2023.

[26] Haoran Que, Jiaheng Liu, Ge Zhang, Chenchen Zhang, Xingwei Qu, Yinghao Ma, Feiyu Duan,

Zhiqi Bai, Jiakai Wang, Yuanxing Zhang, et al. D-cpt law: Domain-specific continual pre-training scaling law for large language models. *arXiv preprint arXiv:2406.01375*, 2024.

[27] Jupinder Parmar, Shrimai Prabhumoye, Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Dan Su, Chen Zhu, Deepak Narayanan, Aastha Jhunjhunwala, Ayush Dattagupta, Vibhu Jawa, Jiwei Liu, Ameya Mahabaleshwarkar, Osvald Nitski, Annika Brundyn, James Maki, Miguel Martinez, Jiaxuan You, John Kamalu, Patrick LeGresley, Denys Fridman, Jared Casper, Ashwath Aithal, Oleksii Kuchaiev, Mohammad Shoeybi, Jonathan Cohen, and Bryan Catanzaro. Nemotron-4 15b technical report, 2024. URL https://arxiv.org/abs/2402.16819.

[28] Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1m dollars, 2024. URL https://arxiv.org/abs/2404.07413.

[29] Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

[30] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

[31] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*, 2023.

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

[33] 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, 2024. URL https://arxiv.org/abs/2309.12284.

[34] Chengpeng Li, Zheng Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. Query and response augmentation cannot help out-of-domain math reasoning generalization, 2023.

[35] 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, 2021. URL https://arxiv.org/abs/2104.04473.

[36] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[38] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. ArXiv, May 2020. URL https://www.microsoft.com/en-us/research/publication/zero-memory-optimizations-toward-training-trillion-parameter-models/.

[39] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling

training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

[40] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

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

[42] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

[43] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese, 2024. URL https://arxiv.org/abs/2306.09212.

[44] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[45] 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. *NeurIPS*, 2021.

[46] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

[47] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

[48] Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

[49] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[50] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

[51] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

[52] Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024.

[53] Xihe Qiu, Teqi Hao, Shaojie Shi, Xiaoyu Tan, and Yu-Jie Xiong. Chain-of-lora: Enhancing the instruction fine-tuning performance of low-rank adaptation on diverse instruction set. *IEEE Signal Processing Letters*, 2024.

[54] Dongfu Jiang, Yishan Li, Ge Zhang, Wenhao Huang, Bill Yuchen Lin, and Wenhu Chen. Tigerscore: Towards building explainable metric for all text generation tasks. *Transactions on Machine Learning Research*, 2023.

[55] 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.

[56] 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.

[57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[58] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[59] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[60] 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.

[61] Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Zhuoer Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, Xiaohan Zhang, Lichao Sun, Hongning Wang, Jing Zhang, Minlie Huang, Yuxiao Dong, and Jie Tang. Alignbench: Benchmarking chinese alignment of large language models, 2023.

[62] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024. URL https://lmsys.org/blog/2024-04-19-arena-hard/.

[63] 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.

[64] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

[65] Qwen Team. Introducing qwen1.5, February 2024. URL https://qwenlm.github.io/blog/qwen1.5/.

[66] gkamradt. Llmtest needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

[67] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomás Mikolov. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651, 2016. URL http://arxiv.org/abs/1612.03651.

[68] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446.

[69] Hugging Face. Blogpost on fineweb v1, 2024. URL https://huggingface.co/spaces/HuggingFaceFW/blogpost-fineweb-v1.

[70] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

[71] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[72] Guohai Xu, Jiayi Liu, Ming Yan, Haotian Xu, Jinghui Si, Zhuoran Zhou, Peng Yi, Xing Gao, Jitao Sang, Rong Zhang, Ji Zhang, Chao Peng, Fei Huang, and Jingren Zhou. Cvalues: Measuring the values of chinese large language models from safety to responsibility, 2023.

[73] Ge Zhang, Yemin Shi, Ruibo Liu, Ruibin Yuan, Yizhi Li, Siwei Dong, Yu Shu, Zhaoqun Li, Zekun Wang, Chenghua Lin, et al. Chinese open instruction generalist: A preliminary release, 2023.

[74] Zhenting Qi, Xiaoyu Tan, Chao Qu, Yinghui Xu, and Yuan Qi. Safer: a robust and efficient framework for fine-tuning bert-based classifier with noisy labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 390–403, 2023.

[75] Xiaoyu Tan, Lin Yong, Shengyu Zhu, Chao Qu, Xihe Qiu, Xu Yinghui, Peng Cui, and Yuan Qi. Provably invariant learning without domain information. In *International Conference on Machine Learning*, pages 33563–33580. PMLR, 2023.

[76] Xiaoyu Tan, Leijun Cheng, Xihe Qiu, Shaojie Shi, Yuan Cheng, Wei Chu, Yinghui Xu, and Yuan Qi. Enhancing task performance in continual instruction fine-tuning through format uniformity. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2384–2389, 2024.

[77] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/

dolly-first-open-commercially-viable-instruction-tuned-llm.

[78] Yuelin Bai, Xinrun Du, Yiming Liang, Yonggang Jin, Ziqiang Liu, Junting Zhou, Tianyu Zheng, Xincheng Zhang, Nuo Ma, Zekun Wang, et al. Coig-cqia: Quality is all you need for chinese instruction fine-tuning, 2024.

[79] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[80] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

[81] Zhenting Qi, Xiaoyu Tan, Shaojie Shi, Chao Qu, Yinghui Xu, and Yuan Qi. Pillow: Enhancing efficient instruction fine-tuning via prompt matching. *arXiv preprint arXiv:2312.05621*, 2023.

[82] Xiaoyu Tan, Shaojie Shi, Xihe Qiu, Chao Qu, Zhenting Qi, Yinghui Xu, and Yuan Qi. Self-criticism: Aligning large language models with their understanding of helpfulness, honesty, and harmlessness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 650–662, 2023.

[83] Tianyu Zheng, Shuyue Guo, Xingwei Qu, Jiawei Guo, Weixu Zhang, Xinrun Du, Chenghua Lin, Wenhao Huang, Wenhu Chen, Jie Fu, et al. Kun: Answer polishment for chinese self-alignment with instruction back-translation. *arXiv preprint arXiv:2401.06477*, 2024.

[84] 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. *arXiv preprint arXiv:2402.14658*, 2024.

[85] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[86] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[87] Ruibin Yuan, Hanfeng Lin, Yi Wang, Zeyue Tian, Shangda Wu, Tianhao Shen, Ge Zhang, Yuhang Wu, Cong Liu, Ziya Zhou, et al. Chatmusician: Understanding and generating music intrinsically with llm. *arXiv preprint arXiv:2402.16153*, 2024.

[88] Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W Huang, Jie Fu, Xiang Yue, and Wenhu Chen. Structlm: Towards building generalist models for structured knowledge grounding. *arXiv preprint arXiv:2402.16671*, 2024.

[89] Qixin Deng, Qikai Yang, Ruibin Yuan, Yipeng Huang, Yi Wang, Xubo Liu, Zeyue Tian, Jiahao Pan, Ge Zhang, Hanfeng Lin, et al. Composerx: Multi-agent symbolic music composition with llms. *arXiv preprint arXiv:2404.18081*, 2024.

[90] Yukang Chen, Shaozuo Yu, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Long alpaca: Long-context instruction-following models. https://github.com/dvlab-research/LongLoRA, 2023.

[91] Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. Longalign: A recipe for long context alignment of large language models, 2024. URL https://arxiv.org/abs/2401.18058.

[92] Wenbo Pan. Fusang-v1: A large curation of instruction-tuning datasets for better bilingual and long-range llms, 2024. URL https://huggingface.co/datasets/wenbopan/Fusang-v1.