FineInstructions: Scaling Synthetic Instructions to Pre-Training Scale

Ajay Patel*
University of Pennsylvania

Colin Raffel
University of Toronto
Vector Institute
Hugging Face

Chris Callison-Burch
University of Pennsylvania

Abstract

Due to limited supervised training data, large language models (LLMs) are typically pre-trained via a self-supervised "predict the next word" objective on a vast amount of unstructured text data. To make the resulting model useful to users, it is further trained on a far smaller amount of "instruction-tuning" data comprised of supervised training examples of instructions and responses. To overcome the limited amount of supervised data, we propose a procedure that can transform the knowledge in internet-scale pre-training documents into billions of synthetic instruction and answer training pairs. The resulting dataset, called FineInstructions, uses ~18M instruction templates created from real user-written queries and prompts. These instruction templates are matched to and instantiated with human-written source documents from unstructured pre-training corpora. With "supervised" synthetic training data generated at this scale, an LLM can be pre-trained from scratch solely with the instruction-tuning objective, which is far more in-distribution with the expected downstream usage of LLMs (responding to user prompts). We conduct controlled token-for-token training experiments and find pre-training on FineInstructions outperforms standard pre-training and other proposed synthetic pre-training techniques on standard benchmarks measuring free-form response quality. Our resources can be found at https://huggingface.co/fineinstructions.

1 Introduction

During self-supervised pre-training, LLMs are trained using a language modeling task like next token prediction over a large amount of text data. This stage of training is where models acquire the vast majority of their knowledge and where the majority of compute, resources, and time is spent (Raffel et al., 2020; Brown et al., 2020; Kaplan et al., 2020). A pre-trained LLM can then be adapted to have better instruction-following capabilities by being further trained on a relatively small amount of supervised instruction-answer examples in a process known as instruction-tuning (Ouyang et al., 2022; Wei et al., 2021; Sanh et al., 2021; Mishra et al., 2022). Existing instruction-tuning datasets have various issues. Many are relatively small, consisting of a few thousand examples (Conover et al., 2023; Rajani et al., 2023). Others are narrow and unrealistic, consisting of academic NLP tasks converted into instruction-tuning formats with a relatively small number of task templates (Sanh et al., 2021; Wei et al., 2021; Mishra et al., 2022). Frontier language models have been used to generate large quantities of more diverse instruction-answer examples, but this has been shown to ultimately only help mimic those models superficially through distillation (Taori et al., 2023; Mukherjee et al., 2023; Honovich et al., 2022; Gudibande et al., 2023). These issues limit the instruction-tuning stage of LLMs, making it primarily useful for helping the model learn to follow instructions and learn response styles. Consequently, the self-supervised pre-training stage is responsible for encoding the vast majority of knowledge in the model weights (Zhou et al., 2023; Ghosh et al., 2024; Hewitt et al., 2024).

*Corresponding author: patel.ajay285@gmail.com

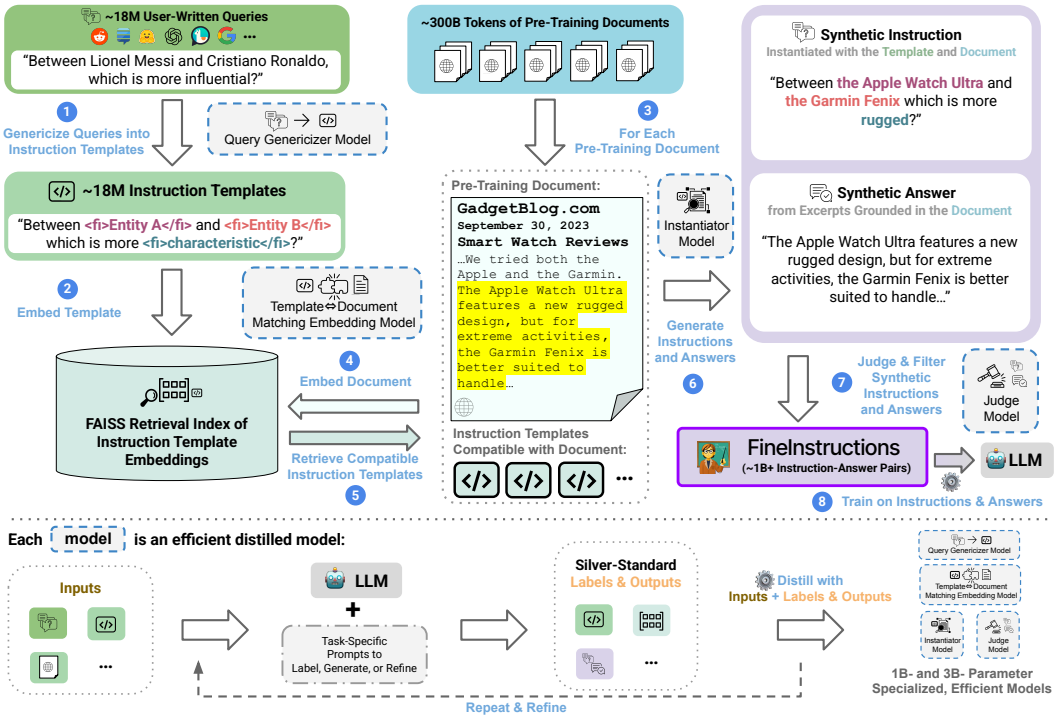


Figure 1: The FineInstructions pipeline for efficiently generating diverse, pre-training scale, synthetic instruction-answer pairs.

Beyond encoding knowledge, pre-training corpora have been shown to help models perform tasks via indirect supervision from examples of tasks that appear in the pre-training documents (Chen et al., 2024). It is unclear, however, whether predicting the next token over pre-training documents is the most optimal or efficient way for models to absorb such capabilities. Recently proposed synthetic rephrasing and transformation pipelines (Maini et al., 2024; Su et al., 2024) demonstrate transforming the original pre-training documents into alternative formats can improve both efficiency of knowledge absorption during pre-training and the performance and capabilities of the trained models.

In this paper, we introduce a procedure called FineInstructions (illustrated in Figure 1) that transforms pre-training corpora into large collections of diverse and realistic instruction-response pairs. Our pipeline pairs documents with instructions or queries a user might ask about the knowledge in the document and then extracts an answer or response grounded in the document. Instructions are created by templatizing ~18M queries written by real users which are then instantiated on a per-document basis. Converting pre-training documents into these synthetic instruction and answer pairs allows us to effectively perform supervised instruction-tuning at pre-training scale. We hypothesize such a restructuring of the data may aid knowledge absorption and model performance as prior studies have shown multi-task datasets and instruction-answer formats induce generalization and instruction-following capabilities (Raffel et al., 2020; Ouyang et al., 2022; Hewitt et al., 2024). This kind of restructuring also prevents pre-training compute budget from being wasted on fitting to potentially low-quality noise in documents (footers, headers, less educational content, etc.) and better utilized towards higher-quality, more educational sections of a document. Finally, it brings pre-training data further in-distribution with the expected downstream usage by users. Ultimately, we demonstrate that the synthetic instruction-answer pairs generated by our approach significantly differ from prior approaches in diversity, complexity, and quality. In sum, our contributions are:

1. We created a large-scale dataset of ~18M instruction templates. The templates were created by mining real user-written queries and tasks and converting them into

generic templates. This large set of templates is what allows us to generate highly diverse and in-distribution synthetic data representative of real user tasks.

2. We introduce the FineInstructions procedure for converting pre-training documents into synthetic instructions and answers at scale.
3. We demonstrate pre-training solely on FineInstructions outperforms standard pre-training and other proposed synthetic transformation pipelines on three LLM evaluation benchmarks correlated with human judgements of model response quality that span academic tasks and more realistic user tasks.
4. We release our code, trained models, and FineInstructions, a dataset of 1B+ synthetic instruction and answer pairs, useful for training LLMs.

2 Related Work

While LLMs are primarily trained on unstructured text sourced from web pages, books, and other natural sources, a number previous works have explored incorporating synthetic training data generated by an LLM itself as part of the pre-training process (Gunasekar et al., 2023; Ben Allal et al., 2024; Abdin et al., 2024; Team et al., 2025; Yang et al., 2025). Other approaches have looked at filtering documents (Penedo et al., 2023; 2024; Li et al., 2024b; Su et al., 2024) with LLM judgments of quality or re-weighting and re-sampling the mixture of documents (Yadlowsky et al., 2023; Albalak et al., 2023; Xie et al., 2023; Li et al., 2024b; Ye et al., 2025; Wettig et al., 2025). Recent approaches have applied more sophisticated techniques, converting raw pre-training documents into instruction-answer pairs and other task formats (Cheng et al., 2024; Su et al., 2024; Yuan & Liu, 2022) or rephrasing low-quality documents into a higher-quality, clean documents (Maini et al., 2024; Su et al., 2024; Nguyen et al., 2025). In our work, we generate synthetic training data by automatically creating and instantiating instruction templates. Other work has explored synthetically generating instruction-tuning data and reasoning data with LLMs and prompting (Honovich et al., 2022; Wang et al., 2022a; Taori et al., 2023; Li et al., 2024c;a; Lambert et al., 2024b; Mukherjee et al., 2023; Lian et al., 2023; Wang et al., 2023a; Jaech et al., 2024; Guo et al., 2025). Köksal et al. (2024) generate instruction-answer pairs for long-form generation, pairing pre-training documents with synthetic instructions that would plausibly generate them. Using instruction templates is also a common approach to creating instruction-tuning datasets (Bach et al., 2022; Sanh et al., 2021; Wei et al., 2021; Mishra et al., 2022; Wang et al., 2022b; Narayan et al., 2024). Approaches that simply prompt LLMs to transform or rephrase documents without templates can yield challenges with diversity in the synthetic data (Ge et al., 2024; Yu et al., 2023). Existing approaches using instruction templates typically use handcrafted or crowdsourced templates, resulting in a far smaller number of templates (hundreds to thousands), lower diversity, and fewer training instances compared to our work.

3 FineInstructions

In this section, we detail the implementation of the FineInstructions pipeline illustrated in Figure 1. Examples of generated outputs can be found in Appendix B. The FineInstructions pipeline takes as input a collection of user-written queries (questions, instructions, and or task requests). These queries are transformed into generic, reusable instruction templates that can be instantiated into a queries for many different subjects, topics, or domains. Given a set of pre-training documents, we match documents with compatible instruction templates. A document is considered a match if it contains enough information to both instantiate the query realistically as well as provide a grounded answer. Finally, we make use of a judge model to measure the quality of these instruction-answer pairs and filter down to a high-quality subset.

Our pipeline enables the creation of a large set of primarily human-written instruction-answer pairs that can be used to perform *supervised* pre-training by conditioning answer generation on each instruction. This procedure closely resembles a weak supervision procedure where large, unlabeled corpora are transformed into large, supervised datasets

with programmatic labeling (Ratner et al., 2017). In this case, we use a large bank of instruction templates and existing models trained on some amount of supervised data to aid in extracting a larger quantity of supervised data. We use DataDreamer (Patel et al., 2024), a framework for synthetic data generation and training, to generate silver-standard data with task-specific prompts and train efficient distilled models that help perform each step of the pipeline at pre-training scale. An example of a task-specific prompt can be found in Appendix D. Full details on our pipeline, prompts, training, and hyperparameters can be found in our code within the supplementary materials. The LLM we use to generate silver-standard data is Llama-3.3 70B Instruct (AI, 2024b). We discuss the implementation of each stage of this pipeline in greater detail in the following sections.

3.1 Generating Instruction Templates

We begin by collecting realistic instances of users querying LLMs, asking questions online (e.g., on forums), or searching via search engines. We also incorporate human-written prompt templates from popular prompt libraries. In total, we source ~21.5M queries across a wide range of categories, including real-world LLM chat logs and search engine queries (e.g., WildChat, LMSys, GooAQ), community QA forums (e.g., StackExchange, Reddit QA), domain-specific datasets (e.g., ChatDoctor), and standard instruction-tuning distributions (e.g., FLAN, Dolly). A comprehensive breakdown of all 20 data sources, including their sizes and references, is provided in Appendix A. After collection, we filter out harmful queries using the OpenAI Moderation API (Markov et al., 2023) and decontaminate from overlap with common benchmarks following Tulu 3 (Lambert et al., 2024a), applying AI2's open-instruct scripts to index source queries and generated pairs in Elasticsearch and remove n-gram matches to benchmark test sets.

We convert queries to genericized templates by inserting `<fi></fi>` tags in place of spans in the query that refer to specific entities or scenarios. Within each pair of tags is a short, natural language description of what kind of text might be appropriate at that span. These spans are filled in with content from a matched document later in our pipeline. To train a model to convert queries into templates, we first select a subset of ~50K queries spanning the selected datasets. These queries are converted to "silver" synthetic templates by prompting an LLM with a series of prompts. We also generate synthetic "compatible document descriptions" for each of these ~50K templates. The description is a short natural language paragraph describing the kinds of documents that could instantiate such a template as well as provide an answer. We train the efficient Query Genericizer Model by fine-tuning a Llama-3.2 1B Instruct model on the generated synthetic data, which is then used to transform all ~18M queries into templates and compatible document descriptions. This pipeline ultimately produces ~18M templates.

3.2 Matching Documents to Instruction Templates

We next use a semantic similarity embedding model, BGE-M3, to embed the compatible document descriptions for the ~18M templates and build a FAISS retrieval index on the resulting embeddings (Xiao et al., 2023; Douze et al., 2025). Given a corpus of unstructured documents (e.g. a representative LLM pre-training dataset), we prompt an LLM to convert ~200K documents into descriptions specifying the type of knowledge contained in each document, which are then embedded with the same model. With each document's embedding, we query the FAISS index and retrieve five potentially compatible instruction templates. We use the LLM to judge whether the document and instruction template are compatible to produce hard positive (1) and hard negative (0) examples of compatibility used with a cosine similarity loss to fine-tune the BGE-M3 embedding model. We fine-tune this model to directly embed documents and compatible document descriptions of instruction templates close together or far apart based on the compatibility labels (Reimers & Gurevych, 2019). To further improve the task-relevance of the embeddings, we repeat this process, ultimately fine-tuning BGE-M3 twice. In the second fine-tuning stage, we introduce a custom pooling layer (described below) that allows us to efficiently produce multiple embeddings at the pooling layer when embedding a single document. Each of the K pooled embeddings aims

to represent each of K different sections or chunks of the document. This allow us to retrieve templates relevant to different pieces of information throughout a long document.

Gaussian Pooling for Document Coverage To ensure retrieved instruction templates adequately cover the entire document, we augment the standard global mean pooling with a custom "Gaussian pooling" layer. Instead of relying solely on a single global embedding, this layer computes multiple "local" embeddings using soft, Gaussian-weighted pooling. By evenly distributing $K = 5$ Gaussian kernels across the text sequence, each embedding captures a distinct semantic region (or "chunk") of the document, yielding 6 total retrieval representations per document (including the global embedding).

In the second fine-tuning phase, we train the embedding model to reduce representation mixing and maintain strict local semantics. We achieve this by applying Gaussian weights to the input attention mask and training on hard positive and negative instruction templates explicitly labeled by their relevant document chunk index. This forces the pooling layer to reliably isolate local representations. Ultimately, this approach is highly effective: in our final FineInstructions dataset, we observe a Pearson correlation of 0.99 between the retrieval chunk index and the actual location of the answer excerpt within the source document (with most excerpts selected from the 19%–71% range, scaling linearly with the chunk index). We defer the rigorous mathematical formulation, pooling operations, and hyperparameter details to Appendix C.

Retrieval When performing retrieval, we consider a template to be a match candidate for a given document if the cosine similarity between their embeddings is above a manually selected threshold (0.865). Among possible matches, we perform a weighted random sample to encourage diversity in the templates: Since our set of instruction templates contain both shorter, simple templates (with only one or two `<fi></fi>` tags) as well as longer, complex templates (with 10+ `<fi></fi>` tags), we compute sampling weights to match the distribution of the complexity of templates derived from real-world LLM queries in the WildChat, LMSys∗, OAsst1, and ShareLM datasets.

3.3 Generating Instructions and Answers

Finally, with a sample of ~100K pre-training documents with six compatible instruction templates each, we prompt an LLM to instantiate the instruction templates and identify excerpt(s) of the document that could be a relevant answer for the query. Since most documents are not written in the style of query responses, we allow the LLM to rephrase the excerpt slightly or add a few words at the beginning of the excerpt that more directly answers the query or instruction before providing more context and detail with an excerpt from the document. Since answers grounded in the document are more desirable than synthetically generated tokens (Shumailov et al., 2023), we ensure that the ratio of excerpted text in generated answers is $\geq$0.80. Finally, we note that generating long responses in this stage is expensive since it involves token-by-token generation as well as possible GPU memory bottlenecks. We additionally note that 1) our answers are mainly direct excerpts from the document and 2) variables in the instruction template are mainly filled in by direct excerpts from the document. This allows us to reduce the computation required during synthetic data generation by introducing tags that designate that text should be directly copied instead of generated. For example, the text "It is known that no preferred inertial frame exists according to the principle of relativity" is generated simply as "`<excerpt>`It is known that`<...>`the principle of relativity.`</excerpt>`". In this way, the model only needs to generate `<excerpt>` tags with ellipsis notation (`<...>`) to reduce the number of generated tokens needed to excerpt long spans of text. These generated tags can be expanded inexpensively and programatically after decoding.

We create ~100K silver-standard examples and train a distilled model to instantiate instruction templates and generate answers by fine-tuning Llama-3.2 3B Instruct. With this efficient distilled model, we generate many instruction-answer pairs for a large number of documents and filter for high-quality instantiations and answers with judge prompts using the LLM until we are able to produce ~100K new high-quality instantiated instruction

and answer examples. We keep some instances of when the LLM determines a retrieved instruction template is not compatible with the document as examples of when to output `null` instead of force a low-quality generation (~5% of the total examples). We train the final Instantiator Model in a second round with these new examples. We ensure these ~100K new examples are also well stratified in length, complexity (number of template variables), and in topic using keyword-based filters to identify instruction-answer pairs related to math and code to ensure balanced representation in the distillation examples.

3.4 Judging and Filtering Instructions and Answers

Having pre-training data formatted as standalone instruction-answer pairs makes our generated data amenable to being used with off-the-shelf reward and judge models. We implement a judging and filtering stage on our synthetic instructions and answers to yield a higher-quality set. Specifically, we use the off-the-shelf Flow Judge model, a distilled 3.8B parameter judge model (AI, 2024a) with a 5-point Likert scale rubric ranging from 1 (irrelevant or off-topic) to 5 (addresses the query without extraneous, vague, or repetitive content), and retain instruction-answer pairs that score ≥ 4. In practice, the judge also filters malformed template instantiations and answers not grounded in the source document.

4 Experimental Setup

In this section, we describe our experimental setup to validate whether training from scratch on synthetic instruction and answer pairs from the FineInstructions pipeline improves knowledge absorption and model performance.

4.1 Baselines

For fair comparison, all methods we evaluate utilize the same unstructured document source corpora and all compared models are trained on the same number of tokens. As a first baseline, we consider training on the original documents themselves, as is done in standard pre-training pipelines. We also select a number of relevant baselines from prior work that propose similar procedures of converting pre-training documents with synthetic transformations or rephrasing. The authors of these prior works have released the vanilla pre-training corpora used in their experiments as well as their pre-computed synthetic transformations over the corpora. We describe them below.

Instruction Pre-Training (IPT) The "Instruction Pre-Training" method (Cheng et al., 2024) proposes training an instruction-and-response synthesizer model to convert documents into instruction-answer pairs. The synthesizer model is trained on instructions and responses from academic NLP Q&A datasets. We use both the IPT vanilla pre-training corpus consisting of ~23B tokens from RefinedWeb (Penedo et al., 2023) and the pre-computed synthetically transformed version of that corpus converted via the IPT instruction-and-response synthesizer model.

Nemotron-CC The Nemotron-CC method (Su et al., 2024) generates synthetic pre-training data by processing high-quality documents filtered from CommonCrawl (Foundation, 2025) with an LLM using a mixture of tasks. These tasks include rephrasing documents, generating synthetic Q&A pairs, and extracting, distilling, and listing core knowledge from the document. We use the Nemotron-CC vanilla pre-training corpus (~300B tokens) and the pre-computed Nemotron-CC synthetic pre-training data created using their mixture of tasks. We also select ~300B tokens of pre-training data generated solely by their method of generating diverse Q&A as a standalone baseline that is the most directly comparable to IPT and FineInstructions. Lastly, we select ~300B tokens of their pre-computed synthetically rephrased pre-training data implementing the WRAP (Web Rephrase Augmented Pre-training) technique (Maini et al., 2024) to compare with a strong rephrasing baseline.

4.2 Pre-Training

Using the original unstructured documents from both datasets, we use our pipeline to retrieve six instruction templates (with $K = 5$ Gaussian pooling chunks to cover the document) and generate six instruction-answer pairs per document. For each pre-training document, we randomly keep instruction-answer pairs with a total token count that does not exceed the token count of the source document.[1] On average, this results in ~3 instruction-answer pairs per document. We format these instruction and answer pairs with a simple chat template similar to the other synthetic baselines: "Instruction: `{{instruction}}\n\n`Answer: `{{answer}}`" (included in the token count). This allows us to produce a dataset-controlled, token-for-token, equivalent set of FineInstructions pre-training data that is ~23B tokens and ~300B tokens for the IPT and Nemotron-CC datasets respectively. Distinct from FineInstructions, the baseline methods do not produce standalone question-answer pairs and instead produce questions that only make sense when the source document is provided as context (e.g. "What club does Helen like?"). They typically append their question-answer pairs at the end of the document (reading comprehension style Q&A) and we pre-train them in their native format. We perform our pre-training experiments using the Lingua framework (Videau et al., 2024) designed for controlled pre-training ablations on 8xH100s and pre-train 1.8B parameter models with a Llama-3 tokenizer (AI, 2024b) for each vanilla pre-training dataset, each baseline dataset, and our FineInstructions datasets. We train for a single epoch on datasets derived from Nemotron-CC and four epochs on datasets derived from IPT.

4.3 Benchmarks

We select three LLM evaluation benchmarks described below that aim to correlate with human judgments of model response quality. These benchmarks evaluate both knowledge absorption as well as the ability to respond to realistic user queries about recommendations, advice, suggestions, etc. When evaluating each method, we format the benchmark questions and instructions into a chat template that matches the method's training template and use the standard ("Instruction:" and "Answer:") template for models pre-trained on vanilla pre-training data. We use greedy sampling when generating responses. Manually inspecting the responses for each method, we find all methods are able to respond and format answers to prompts reasonably well for accurate judging with few degenerate responses.

MixEval The MixEval benchmark (Ni et al., 2024) is a subset of task instances from a variety of academic LLM benchmarks such as TriviaQA (Joshi et al., 2017), MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), CommonsenseQA (Talmor et al., 2019), among others. MixEval aims to retain only examples from each of these datasets that best correlate with human judgments of response quality and grades responses with an LLM-as-judge given a reference answer. We use GPT-5 mini as a judge (OpenAI, 2025) and use the "2024-08-11" version on both the "Standard" and "Hard" splits.

MT-Bench-101 MT-Bench-101 is a multi-turn benchmark involving realistic, subjective user queries (opinions, advice, writing, etc.) graded via LLM-as-judge with a rubric scaled out of 10 (Bai et al., 2024). While it is a multi-turn benchmark, it is common to evaluate on this benchmark in single-turn fashion for models not tuned for multi-turn chat. We use GPT-5 mini as a judge (OpenAI, 2025).

AlpacaEval The AlpacaEval benchmark evaluates on realistic queries and tasks a user may ask a LLM and evaluates with LLM-as-judge in a head-to-head evaluation between the responses from two models to produce a win rate (Li et al., 2023a). It also corrects for length-bias in judging (Dubois et al., 2024). We use the supported GPT-4-Turbo model as our judge model (OpenAI, 2023).

[1] If there is remaining token budget because instruction-answer pairs cannot exactly fill the token count of the source pre-training document, we rollover the remaining token budget to instruction-answer pairs for future pre-training documents.

5 Results

Method	MixEval *Acc (%)*		MT-Bench-101 *Likert Score*	AlpacaEval *FI Win Rate % (Δ Win Margin %)*
	Standard	**Hard**		
IPT Corpus (23B)				
Standard Pre-Training	17.8	14.0	1.9	**73.6%** (Δ **47.2**%)
IPT	19.8	16.7	2.4	**68.2%** (Δ **36.4**%)
FineInstructions	**31.7**	**19.2**	**2.8**	–
Nemotron-CC Corpus (300B)				
Standard Pre-Training	24.0	17.1	3.5	**63.6%** (Δ **27.2**%)
WRAP	22.8	18.4	3.6	**65.1%** (Δ **30.2**%)
Q&A	27.1	18.9	3.4	**76.1%** (Δ **52.2**%)
Nemotron-CC	24.5	16.7	3.6	**65.9%** (Δ **31.8**%)
FineInstructions	**33.0**	**21.8**	**3.9**	–

Table 1: Benchmark performance of 1.8B parameter models pre-trained on FineInstructions and various baselines. We **bold** the strongest method and **bold** the win rate % if FineInstructions wins.

The results of our pre-training experiments can be found in Table 1. Pre-training on data from the FineInstructions pipeline outperforms both standard pre-training and the other synthetic baselines on both datasets. For example, we observe a ~69% relative improvement on MixEval compared to standard pre-training on the IPT dataset and a ~39% improvement on Nemotron-CC. A per-sub-benchmark breakdown of MixEval and MixEval-Hard performance is provided in Appendix H. On AlpacaEval, outputs from training on FineInstructions are consistently preferred over outputs from all other pipelines. MT-Bench-101 is the only reference-free evaluation (with no gold reference answer or head-to-head comparison) resulting in evaluation scores that yield lower spread and differentiation between models. Nevertheless, our FineInstructions data achieves a higher score on MT-Bench-101 than any other method. Notably, these improvements hold across both knowledge-focused (MixEval) and open-ended evaluation (MT-Bench-101 and AlpacaEval) benchmarks, suggesting that FineInstructions yields more consistent generalization across both kinds of tasks. Both IPT and Nemotron-CC demonstrated fairly small improvement (0-2%) on MixEval despite generating narrow instruction data in a highly similar style targeting the evaluation benchmark format (multiple choice, reading comprehension Q&A, etc.). On more realistic evaluations correlated to human judgments of response quality on open-ended real-world tasks such as those in AlpacaEval, we find these other techniques yield subpar results compared with FineInstructions. We hypothesize this is supported by the relative diversity of queries in FineInstructions compared to other pipelines' focus on generating instructions for a narrow set of academic benchmark tasks like multiple choice and Q&A. Finally, in Appendix G, we experiment with pre-training models at various sizes (300M, 1.8B, and 7B parameters) on FineInstructions and the strongest-performing baseline (Nemotron-CC) and find training on FineInstructions reaches higher performance levels with equivalent token and compute budget, making it an optimal technique for producing data for pre-training capable, small, and efficient language models.

6 Discussion

In this section, we provide an analysis on the diversity of the generated instructions, discuss an additional experiment on ablating the judging stage, and sketch out future directions.

6.1 Diversity of Instructions

We analyze our generated instructions to ensure there is sufficient diversity with no over-representation of certain instruction templates. It is desirable that long-tail instruction tem-

plates (e.g. "Give me free apps that convert a file from <fi>format #1</fi> to <fi>format #2</fi>") that are only compatible with highly specific documents are represented along with simpler instruction templates compatible with many documents (e.g. "Give me a summary of <fi>topic</fi>"). When applied to the base Nemotron pre-training dataset, 4.3M unique instruction templates were used to instantiate ~1.08B total instructions. We find no single instruction template comprises more than 0.09% of the generated instructions and that the majority of instruction templates are used to instantiate fewer than 1,000 instructions. Consequently, our data is highly diverse in task and task formats. Overall, we find a power-fit relationship between the number of pre-training documents (x) and the number of unique instruction templates (y) utilized at least once when generating instructions from those documents from our set of ~18M instruction templates as approximately: $y = 16,891 * x^{0.24}$ with an $r^2 = 0.96$. Around ~50% of instructions are instantiated from a template derived from GooAQ, with Reddit QA (~27%), LMSys Chat (~9%), and WildChat (~6%) following behind. All other sources represent $\leq 1\%$ each.

We also analyze the type of the instructions generated. We use Llama-3.3 70B Instruct with zero-shot prompts to classify instruction templates into various non-mutually exclusive domains (science, math, code, medicine, etc.) and provide the results in Table 5 in Appendix E. Additionally, we define categories for measuring whether an instruction is a task that requires reasoning over knowledge (e.g. "Compare and contrast <fi>entity A</fi> and <fi>entity B</fi>") as well as if the instruction is "tasky" (e.g. "Write a critique of <fi>media performance</fi>") vs. a simple knowledge recall Q&A task. Finally, we use the LLM to generate a short, one-sentence description of each instruction template describing the task at hand and visualize these sentences in a sunburst chart for a subset of instructions in Figure 2 in Appendix E following Köksal et al. (2024). This kind of analysis with zero-shot classification may yield errors, but provides some high-level insights into the composition of tasks. The full annotation prompts can be found in our code.

6.2 Effect of Judging and Filtering

We conduct an ablation experiment to measure the effect of the final judging and filtering stage. We pre-train with and without judging and filtering and provide these results in Appendix F on the IPT and Nemotron-CC datasets and find this added stage overall further improves the performance, especially on the AlpacaEval benchmark.

6.3 Limitations and Future Directions

We intentionally do not optimize for or report perplexity, a standard proxy for language modeling ability. Because FineInstructions pre-trains models exclusively on instruction-answer pairs rather than raw next-token prediction over documents, we openly concede that standard pre-training likely outperforms our approach on perplexity-based metrics. Our primary optimization target is downstream user utility (producing useful responses to prompts), rather than intermediate language modeling proxies, and our evaluation suite reflects this objective. Our comparisons against baselines may appear format-mismatched or that our approach has a supervision advantage. However, our approach and the synthetic baselines (IPT, Nemotron-CC, WRAP, and Q&A) operate on the same principle: converting raw pre-training documents into a query-response training format. The performance gains we report over these baselines therefore stem from the design of our pipeline (diverse user-derived templates for more realistic queries, document-grounded excerpt answers, and judge filtering).

The FineInstructions pipeline could be optimized and scaled to yield better results. Several procedural choices, including 50K-example distillation sets and a retrieval threshold of 0.865, reflect compute budget constraints and manual calibration rather than systematic tuning. The distribution of source queries, calibration of the matching embedding, and the sampling weights all have influence on the composition and complexity of instructions generated. An optimal mixture may yield further improvements in performance. One common failure mode we observe in the generated instructions is that complex templates are challenging to match and instantiate. Scaling to larger models beyond the 1B and 3B scales considered in

this work could yield stronger performance on complex templates and longer documents. Although our chosen benchmarks provide a reliable picture of language model performance in a range of scenarios, we also found that there is a paucity of benchmarks targeting the kind of long-tail realistic knowledge tasks users ask LLMs (recommendations, advice, suggestions, etc.) as opposed to factual recall of knowledge. Moreover, pre-training on instruction-answer pairs yields a model that consistently produces long-form answers and assigns a low probability to answer choices or short-form responses. This makes benchmarks using log probability-based classification for evaluation ill-suited for our setting and prior studies have found such evaluation is not reliable (Wang et al., 2024; Chandak et al., 2025). Instead, extractive or LLM-as-judge based grading is recommended, which we follow. There are comparatively few LLM benchmarks in the latter category.

7 Conclusion

FineInstructions provides an effective pipeline for transforming real user queries into templates to generate in-distribution synthetic data at scale. We demonstrate that the resulting data can be effectively used to train LLMs in a supervised, instruction-aligned format rather than through self-supervised next token prediction on pre-training documents. By transforming the learning objective and the structure of pre-training data, this approach trains models on data that better reflects downstream usage patterns and improves knowledge absorption efficiency.

Acknowledgements

We would like to acknowledge the Hugging Face organization and team for providing compute and storage resources for the experiments in this work and providing feedback on scaling the procedure. Particular thanks to Lewis Tunstall, Hynek Kydlíček, and Joel Niklaus for discussions around evaluation and related work. Thank you to our reviewers for their comments in strengthening this paper.

This research is based upon work supported in part by the Defense Advanced Research Projects Agency's (DARPA) SciFy program (Agreement No. HR00112520300) and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 56000026C0019. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Ethics Statement

This paper presents work whose goal is to advance large language model training with synthetically generated data. One positive consequence of our work is that it allows for more efficient training with the synthetic corpus that only needs to be transformed and generated once to yield better results and faster convergence on subsequent training runs. Training on synthetically generated data may amplify biases and errors from the generating model. We mitigate this effect by taking near-exact excerpts from source documents and mainly using the generating model to transform naturally occurring text data into the desired format, rather than generate new content that could contain hallucinated content. However, some marginal risk of systematic data biasing may still remain due to the scale of generation in this work.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4

technical report. *ArXiv preprint*, abs/2412.08905, 2024. URL https://arxiv.org/abs/2412.08905.

Flow AI. Flow judge: An open small language model for llm system evaluations. Technical report, Flow AI, 2024a. URL https://www.flow-ai.com/blog/flow-judge. Model: Flow-Judge v0.1 (3.8B parameters); Apache 2.0 license.

Meta AI. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024b.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. Promptsource: An integrated development environment and repository for natural language prompts, 2022.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454, 2024.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Answer matching outperforms multiple choice for language model evaluation. *ArXiv preprint*, abs/2507.02856, 2025. URL https://arxiv.org/abs/2507.02856.

Harrison Chase. LangChain. Technical report, LangChain, 2022. URL https://github.com/langchain-ai/langchain.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. Parallel structures in pre-training data yield in-context learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8582–8592, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.465. URL https://aclanthology.org/2024.acl-long.465/.

Daixuan Cheng, Yuxian Gu, Shaohan Huang, Junyu Bi, Minlie Huang, and Furu Wei. Instruction pre-training: Language models are supervised multitask learners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2529–2550, 2024.

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. Technical report, Databricks, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. The ShareLM collection and plugin: Contributing human-model chats for the benefit of the community. In Pushkar Mishra, Smaranda Muresan, and Tao Yu (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 167–177, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-253-4. URL https://aclanthology.org/2025.acl-demo.17/.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 2025.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *ArXiv preprint*, abs/2404.04475, 2024. URL https://arxiv.org/abs/2404.04475.

Common Crawl Foundation. Common crawl web corpus. https://commoncrawl.org/, 2025. Accessed: 2025-10-26; dataset release CC-MAIN-2025-08.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, Ramaneswaran S, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, and Dinesh Manocha. A closer look at the limitations of instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 15559–15589, 2024.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *ArXiv preprint*, abs/2305.15717, 2023. URL https://arxiv.org/abs/2305.15717.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv preprint*, abs/2501.12948, 2025. URL https://arxiv.org/abs/2501.12948.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.

John Hewitt, Nelson F Liu, Percy Liang, and Christopher D Manning. Instruction following without instruction tuning. *ArXiv preprint*, abs/2409.14254, 2024. URL https://arxiv.org/abs/2409.14254.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor, 2022. URL https://arxiv.org/abs/2212.09689.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *ArXiv preprint*, abs/2412.16720, 2024. URL https://arxiv.org/abs/2412.16720.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. In *ICML 2023*, pp. 14702–14729. International Machine Learning Society (IMLS), 2023.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

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.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. Gooaq: Open question answering with diverse answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 421–433, 2021.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Minh Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Alexandrovich Glushkov, Arnav Varma Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Julian Mattick. Openassistant conversations - democratizing large language model alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=VSJotgbPHF.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. Longform: Effective instruction tuning with reverse instructions, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, 2024a.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu Nouha Dziri Xinxi Lyu, Yuling Gu Saumya Malik Victoria Graf, Jena D Hwang, et al. Tülu 3: Pushing frontiers in open language model post-training. *ArXiv preprint*, abs/2411.15124, 2024b. URL https://arxiv.org/abs/2411.15124.

Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, et al. Synthetic data (almost) from scratch: Generalized instruction tuning for language models. *ArXiv preprint*, abs/2402.13064, 2024a. URL https://arxiv.org/abs/2402.13064.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024b.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*, 2024c.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023a.

Yunxiang Li, Li Zihan, Kai Zhang, Dan Ruilong, Steve Jiang, and You Zhang. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6), 2023b.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/Open-Orca/OpenOrca, 2023.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14044–14072, 2024.

Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. Expertqa: Expert-curated questions and attributed answers. *ArXiv preprint*, abs/2309.07852, 2023. URL https://arxiv.org/abs/2309.07852.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, pp. 15009–15018, 2023.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

Avanika Narayan, Mayee F Chen, Kush Bhatia, and Christopher Re. Cookbook: A framework for improving llm generative abilities via programmatic data generating templates. *ArXiv preprint*, abs/2410.05224, 2024. URL https://arxiv.org/abs/2410.05224.

Thao Nguyen, Yang Li, Olga Golovneva, Luke Zettlemoyer, Sewoong Oh, Ludwig Schmidt, and Xian Li. Recycling the web: A method to enhance pre-training data quality and quantity for language models. *ArXiv preprint*, abs/2506.04689, 2025. URL https://arxiv.org/abs/2506.04689.

Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures. *ArXiv preprint*, abs/2406.06565, 2024. URL https://arxiv.org/abs/2406.06565.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Gpt-5 system card. Technical report, OpenAI, 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Ajay Patel, Colin Raffel, and Chris Callison-Burch. DataDreamer: A tool for synthetic data generation and reproducible LLM workflows. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3781–3799, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.208. URL https://aclanthology.org/2024.acl-long.208.

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

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. No robots. https://huggingface.co/datasets/HuggingFaceH4/no_robots, 2023.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB endowment. International conference on very large data bases*, volume 11, pp. 269, 2017.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan D. Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng-Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. *ArXiv*, abs/2110.08207, 2021. URL https://api.semanticscholar.org/CorpusID:239009562.

Chufan Shi, Yixuan Su, Cheng Yang, Yujiu Yang, and Deng Cai. Specialist or generalist? instruction tuning for specific NLP tasks. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15336–15348, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.947. URL https://aclanthology.org/2023.emnlp-main.947/.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget, 2023.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. *ArXiv preprint*, abs/2412.02595, 2024. URL https://arxiv.org/abs/2412.02595.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *ArXiv preprint*, abs/2507.20534, 2025. URL https://arxiv.org/abs/2507.20534.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL https://github.com/facebookresearch/lingua.

Xinpeng Wang, Bolei Ma, Chengzhi Hu, Leon Weber-Genzel, Paul Röttger, Frauke Kreuter, Dirk Hovy, and Barbara Plank. "my answer is C": First-token probabilities do not match text answers in instruction-tuned language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7407–7416, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.441. URL https://aclanthology.org/2024.findings-acl.441/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023a.

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

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*, 2022b.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. Helpsteer: Multi-attribute helpfulness dataset for steerlm, 2023b.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. *ArXiv preprint*, abs/2502.10341, 2025. URL https://arxiv.org/abs/2502.10341.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Steve Yadlowsky, Lyric Doshi, and Nilesh Tripuraneni. Pretraining data mixtures enable narrow model selection capabilities in transformer models. *ArXiv preprint*, abs/2311.00871, 2023. URL https://arxiv.org/abs/2311.00871.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto. Synthetic continued pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Weizhe Yuan and Pengfei Liu. restructured pre-training. *ArXiv preprint*, abs/2206.11147, 2022. URL https://arxiv.org/abs/2206.11147.

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*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

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, 2023a.

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, 2023b.

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*, 36:55006–55021, 2023.

A Dataset Sources

Below we detail the comprehensive list of datasets used to source user-written queries we convert into instruction templates.

Dataset	Query Count	Citation
StackExchange Instructions	10.6M	-
WildChat	657K	(Zhao et al., 2024)
LMSys Chat	559K	(Zheng et al., 2023a)
LMSys Chatbot Arena Conversations	26.5K	(Zheng et al., 2023b)
OAsst1	3.92K	(Köpf et al., 2023)
HuggingFace NoRobots	9.49K	(Rajani et al., 2023)
HelpSteer	10K	(Wang et al., 2023b)
Dolly	14.7K	(Conover et al., 2023)
Reddit QA	7.47M	-
GooAQ	3.01M	(Khashabi et al., 2021)
ShareLM	264	(Don-Yehiya et al., 2025)
ExpertQA	1.73K	(Malaviya et al., 2023)
ChatDoctor iCliniq	7.32K	(Li et al., 2023b)
ChatDoctor HealthcareMagic	112K	(Li et al., 2023b)
Awesome ChatGPT Prompts	203	-
Anthropic Prompt Library	64	-
LangChain Prompts	51	(Chase, 2022)
NaturalInstructions	757	(Mishra et al., 2022)
P3	544	(Sanh et al., 2021)
FLAN	3.44K	(Wei et al., 2021)

Table 2: Overview of the data sources, their respective query counts, and citations used in our collection process.

B Examples of FineInstructions

Vanilla Pre-Training Document:

Structure and Synchronicity for Better Charting
Two key characteristics will help you to ensure that your notes communicate not only what you did, but also what you were thinking.
Fam Pract Manag. 2011 Jul-Aug;18(4):15-17.

How many times have you read a medical note that does not make the selection of the diagnosis or treatment clear? Have you ever read your own notes after receiving notice of a malpractice suit and winced at the inconsistencies? Poorly constructed medical notes are a widespread problem. I've seen it while reviewing the charts of medical students and residents to ensure that they met the standard of care and avoided malpractice risk. While most physicians document history and physical data with the required number of CPT elements, few clearly convey a line of reasoning that reveals their clinical thought process. A note that documents a detailed history and moderately complex decision making does not necessarily illuminate why certain decisions were reached or why a particular treatment was justified.

I wanted to find out where in the educational process students learned key documentation concepts, so I did some research that included informally querying medical students from multiple medical schools who were rotating at our residency. None of them could describe learning a formal structure for completing assessments and plans, which is consistent with a study that found only 4 percent of standardized encounters were accurately charted by medical students.1 I also discovered that the 2010 United States Medical Licensing Examination clinical skills exam guide states that students are expected to present a list of differential diagnoses in order of likelihood along with desired evaluations, but no requirements existed for discussing the clinical rationale. As a result of these and other findings, I developed a formal framework, described in this article, to teach residents and students an appropriate way to construct their notes. The initial feedback has been highly positive.

Documenting a confirmed diagnosis

To begin, let's review what should be included when documenting a confirmed diagnosis. Generally, six elements are needed...

Instruction Template	Synthetic Instruction	Synthetic Answer
I'm looking for a template or guidelines for how to write a `<fi>type of document, e.g. technical report, research paper, etc.</fi>`. The `<fi>type of document</fi>` is `<fi>context or field of study</fi>`-related and intended for a `<fi>role or position</fi>` to review the work of the `<fi>task or investigation</fi>` of a `<fi>type of issue or problem</fi>` related to `<fi>field or industry</fi>`.	I'm looking for a template or guidelines for how to write a medical note. The medical note is healthcare-related and intended for a physician to review the work of the clinical investigation of a medical issue related to patient care.	To write a medical note, consider the following guidelines. To begin, let's review what should be included when documenting a confirmed diagnosis. Generally, six elements are needed...

Vanilla Pre-Training Document:

Spot writes: ««your dentist is correct diabetes is often accomnpied by severe periodontal disease and complicated by xerosomia (xerostomia, dry mouth)this results in increased incidence of caries and loss of teeth. also infections»»> Jack writes: ««»» Spot and Jack, I know all of this! No, Jack she didn't see the start of periodontal disease, had the full workup and absolutely no sign of it at all. I had one cavity in a flossing area which prompted the extra flossing advice. My whole point was that with a busy life + DM, extra flossing was going to be low on my list of extras to accomplish. I already floss daily, just not 3-4 times a day and after every meal. BTW, my flossing time is way more than 2-3 minutes Jack. Sooooo, I understand the DANGERS of not flossing more often than nonD folks, and if all of my teeth fall out in 50 years when I'm 80, Jack and Spot can say, "Told you so." (vbg) Jill (Who Is Still Not Flossing 5 times a Day)

——————————————— Do You Yahoo!? Yahoo! Auctions - Buy the things you want at great prices.
——————————————— for HELP or to subscribe/unsubscribe, contact: HELP@insulin-pumpers.org send a DONATION

Instruction Template	Synthetic Instruction	Synthetic Answer

Continued on next page...

what effects does <fi>a medical or psychological condition</fi> have?	what effects does diabetes have?	Diabetes has several effects, including being often accompined by severe periodontal disease and complicated by xerosomia (xerostomia, dry mouth)this results in increased incidence of caries and loss of teeth.

Vanilla Pre-Training Document:
This article is the first part of building a production grade nodejs,graphql and typescript server with auto deployment pipeline. there are tons of articles and tutorials are available to explain how to build a scalable backend solutions. but, what i felt missing was a solution that connects all the dots with a simple and proper use-cases. Most of the time, we won't directly jump into building the application by coding it. we would start with designing how the backend should be based on the application requirement. knowing how to do that is an important skill to have than just knowing the syntax of x or y framework.

Before coding, let's see what we are going to build to learn the concepts. we will be building a Image management application where users can manage their image resources. Mainly, it will contain login,signup and main application. In main application, user can upload images and view all the uploaded images. Even though it's a simple use-case, we can understand all the concepts using this simple use-case.

Login Screen
Main App Screen

Now, we know what application we are going to build. let's apply all the best practices to build a Nodejs,GraphQL and TypeScript Server.

This part is going to explain how to setup the application and create a query and mutations using GraphQL and TypeScript. we will be using type-graphql for your application. let's install the required dependancies for our application, npm i apollo-server-express express graphql reflect-metadata type-graphql mongoose....

Instruction Template	Synthetic Instruction	Synthetic Answer
how to <fi>a few word description of the action to be performed</fi> <fi>name of a service, system, or platform</fi>?	how to create a GraphQL schema and connect to a MongoDB database using type-graphql?	To create a GraphQL schema and connect to a MongoDB database using type-graphql, follow these steps: we will be using type-graphql for your application. let's install the required dependancies for our application, 1npm i apollo-server-express express graphql reflect-metadata type-graphql mongoose...

C Gaussian Pooling Layer for Template ⇔ Document Matching Embedding Model

Specifically, let $H = [h_1, \ldots, h_T]$ denote the sequence of token embeddings with attention mask $m_t \in \{0, 1\}$, where T is the effective sequence length. We define K as the number of Gaussians representing different chunks of the document, c_k as the chunk center positions, $\rho_k = \frac{k}{K+1}$ as their normalized positions, σ as the Gaussian width parameter, and $\alpha \in [0, 1]$ as the blending weight.

In our implementation, we choose $K = 5$ chunks, $\alpha = 1.0$, and $\sigma = 0.05$. Using H, m_t, and T, we compute a global mean embedding and K Gaussian-weighted chunk embeddings centered at $c_k = \rho_k T$ with width σ. Each chunk embedding is optionally blended with the global embedding via α.

During the second round of fine-tuning, to train the model to maintain local semantic representations, we create training instances by applying Gaussian weights at the input attention mask. We only attend to tokens with a weight of ≥ 0.5, producing K distinct

local embeddings with these attention masks to retrieve and compute the loss against hard negative and hard positive examples.

Input Token Embeddings and Attention Mask:

$$H = [h_1, \ldots, h_T],\ h_t \in \mathbb{R}^d, \quad m_t \in \{0, 1\}$$

Global Embedding:

$$\bar{h} = \frac{\sum_{t=1}^{T} m_t h_t}{\sum_{t=1}^{T} m_t}$$

Gaussian Centers for K Chunks:

$$c_k = \rho_k T, \quad \rho_k = \frac{k}{K+1},\ k = 1, \ldots, K$$

Gaussian Weights:

$$w_{k,t} = \frac{m_t \exp\left(-\frac{1}{2}\left(\frac{t - c_k}{\sigma T}\right)^2\right)}{\sum_{t'=1}^{T} m_{t'} \exp\left(-\frac{1}{2}\left(\frac{t' - c_k}{\sigma T}\right)^2\right)}$$

Chunk-Local Embeddings:

$$\tilde{h}_k = \sum_{t=1}^{T} w_{k,t} h_t$$

Blend Chunk-Local Embeddings with Global Embedding:

$$h_k^* = (1 - \alpha)\bar{h} + \alpha \tilde{h}_k$$

Final Global and K Chunk-Local Embeddings:

$$E = [\,\bar{h},\ h_1^*,\ \ldots,\ h_K^*\,] \in \mathbb{R}^{d(K+1)}$$

D Example of Prompt in FineInstructions Pipeline

Below is a generic query template that can be used for instruction-tuning an LLM. It contains a template of a natural language user query, with template variables using <fi></fi> tags ("<fi>description of content that should go there....</fi>").

{{example of a template...}}

These templates can then be instantiated from a *compatible* document (such as a web page, article, book, essay, etc.) that has contain enough information to both fill in all of the template variables and also has the answer to the query so that we can create grounded, realistic, diverse instruction-tuning questions and answers using a combination of the template + document.

Given the provided template + a compatible document we found below:

1. **Task:** Can you instantiate the template by filling in the <fi></fi> template variables to create a realistic instruction that could be realistically asked about the content/topic/subject described in the document?

2. **Answerability and Template Incompatibility:** ...

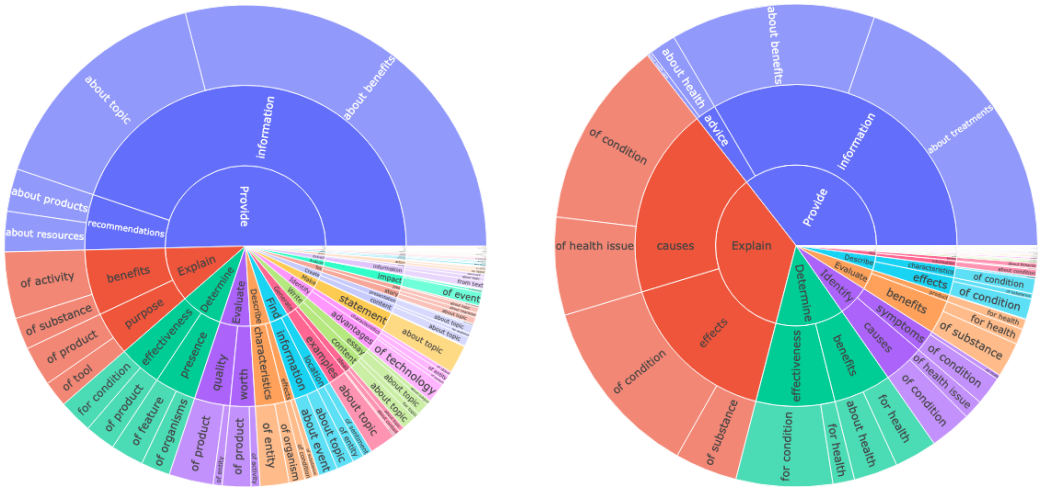
{{other requirements...}}

<<<Instruction Template:
`{{template}}`
>>>

<<<Compatible Document:
`{{document}}`
>>>

E Diversity of FineInstructions

We visualize the diversity of instructions and tasks in FineInstructions below.



(a) FineInstructions (subset) (b) FineInstructions (medical subset)

Figure 2: A visualization of the task diversity in a small subset of FineInstructions. We also preview domain-specific diversity, comparing with a subset that was instantiated from medicine-related templates. Charts can be read from the inner ring to the outer ring for each "slice".

We note datasets like FineInstructions can be used to mine for large amounts of task-specific or domain-specific instructions. For example, with 1B+ instructions, even 0.58% of the instructions being math instructions yields 6M+ math instructions. This could enable experiments on training with different mixtures of topics and task types as well as experiments with increasing the proportion of pre-training data on specific tasks to build a domain-specialist LLM which some prior work has found to be important to downstream performance (Jang et al., 2023; Shi et al., 2023).

Domain/Category	Percentage (%)
Science	36.61%
Math	0.58%
Code	0.25%
Medicine	10.40%
Personal Life	14.74%
Reasoning Task	10.99%
Tasky vs. Q&A	6.42%

Table 5: 1B+ instructions from FineInstructions classified into various categories.

F Ablation of Judging and Filtering Stage

Method	MixEval *Acc (%)*		MT-Bench-101 *Likert Score*	AlpacaEval *FI Win Rate % (Δ Win Margin %)*	
	Standard	**Hard**		**Without Judging**	**With Judging**
IPT Corpus (23B)					
Standard Pre-Training	17.8	14.0	1.9	72.1% (Δ 44.2%)	**73.6%** (Δ **47.2**%)
IPT	19.8	16.7	2.4	61.7% (Δ 23.4%)	**68.2%** (Δ **36.4**%)
FineInstructions	30.1	**20.2**	2.6	–	46.5% (Δ -7%)
FineInstructions (judged)	**31.7**	19.2	**2.8**	–	–
Nemotron-CC Corpus (300B)					
Standard Pre-Training	24.0	17.1	3.5	54.9% (Δ 9.8%)	**63.6%** (Δ **27.2**%)
WRAP	22.8	18.4	3.6	58.4% (Δ 16.8%)	**65.1%** (Δ **30.2**%)
Q&A	27.1	18.9	3.4	68.7% (Δ 37.4%)	**76.1%** (Δ **52.2**%)
Nemotron-CC	24.5	16.7	3.6	55.7% (Δ 11.4%)	**65.9%** (Δ **31.8**%)
FineInstructions	**33.3**	**21.8**	3.6	–	**61.4%** (Δ **22.8**%)
FineInstructions (judged)	33.0	**21.8**	**3.9**	–	–

Table 6: Incorporating a judging and filtering stage on top of FineInstructions leads to further improvements in performance.

G Performance of Training on FineInstructions at Different Model Scales

Method	MixEval *Acc (%)*		MT-Bench-101 *Likert Score*	AlpacaEval *FI Win Rate % (Δ Win Margin %)*	
	Standard	**Hard**		**One Model Size Down**	**Same Model Size**
Nemotron-CC Corpus (300B) – 300M Parameter Models					
Standard Pre-Training	18.5	14.3	2.8	–	**65.1%** (Δ **30.2**%)
Nemotron-CC	18.7	13.9	3.0	–	**57.5%** (Δ **15.0**%)
FineInstructions	**31.9**	**22.0**	**3.2**	–	–
Nemotron-CC Corpus (300B) – 1.8B Parameter Models					
Standard Pre-Training	24.0	17.1	3.5	49.2% (Δ -1.6%)	**63.6%** (Δ **27.2**%)
Nemotron-CC	24.5	16.7	3.6	49.8% (Δ -0.4%)	**65.9%** (Δ **31.8**%)
FineInstructions	**33.0**	**21.8**	**3.9**	–	–
Nemotron-CC Corpus (300B) – 7B Parameter Models					
Standard Pre-Training	33.0	21.6	**4.4**	50.0% (Δ 0.0%)	**59.0%** (Δ **18.0**%)
Nemotron-CC	32.7	21.5	4.3	45.6% (Δ -8.8%)	**56.6%** (Δ **13.2**%)
FineInstructions	**42.6**	**25.8**	4.3	–	–

Table 7: Pre-training results for models with 300M, 1.8B, and 7B parameters. At a fixed model size, models pre-trained on FineInstructions are competitive with or outperform, baselines trained at the next larger scale (e.g., 300M $\rightarrow$ 1.8B, 1.8B $\rightarrow$ 7B). Under equivalent token and compute budgets, FineInstructions consistently achieves higher performance across MixEval, MT-Bench-101, and AlpacaEval, indicating that it is an effective data mixture for training smaller, more compute-efficient models with stronger capabilities.

H Detailed Breakdown of Sub-Benchmarks in MixEval

Table 8 and Table 9 provide a per-sub-benchmark breakdown of MixEval and MixEval-Hard performance for all pre-training methods evaluated in this work. Sample counts (n) for each sub-benchmark are shown in the column headers.

		IPT Corpus (23B)			Nemotron-CC Corpus (300B)				
Sub-Benchmark	n	**Std. PT**	**IPT**	**FI**	**Std. PT**	**WRAP**	**Q&A**	**Nemo-CC**	**FI**
Overall		17.8	19.8	**31.7**	24.0	22.8	27.1	24.5	**33.0**
TriviaQA	1328	3.2	6.7	**13.2**	20.4	16.8	**23.0**	20.3	20.9
MMLU	681	30.1	28.0	**47.0**	28.0	29.7	33.5	27.8	**45.2**
DROP	473	5.2	11.9	**33.2**	13.6	14.0	19.9	13.7	**38.0**
HellaSwag	308	32.8	29.2	**43.5**	23.7	28.6	28.9	36.7	**37.7**
CommonsenseQA	202	30.7	26.2	**48.0**	22.8	22.3	29.7	24.3	**39.1**
MMLU-Pro	185	15.1	16.8	**33.5**	15.7	10.8	14.1	16.2	**25.9**
BoolQ	171	53.8	55.0	**55.6**	56.1	50.9	45.6	48.0	**59.1**
AGIEval	121	14.8	25.9	**28.7**	25.0	**27.8**	26.9	20.4	23.1
BBH	115	10.3	**40.2**	22.8	**25.0**	15.3	16.8	**25.0**	24.3
PIQA	105	41.0	36.2	**56.2**	52.4	54.3	56.2	50.5	**61.9**
SIQA	93	33.3	32.3	**38.7**	**41.9**	35.5	37.6	39.8	31.2
ARC	91	29.7	30.8	**50.5**	20.9	30.8	40.7	26.4	**49.5**
OpenBookQA	43	**51.2**	25.6	44.2	37.2	25.6	32.6	27.9	**48.8**
GSM8k	40	0.0	**1.3**	0.0	0.0	0.0	**2.8**	0.5	1.3
MATH	31	0.0	0.0	0.0	0.0	0.0	**3.2**	0.0	0.0
GPQA	4	**25.0**	0.0	0.0	25.0	0.0	50.0	0.0	**75.0**
WinoGrande	4	0.0	25.0	**75.0**	50.0	25.0	25.0	**75.0**	0.0
MBPP	3	66.7	66.7	**100.0**	66.7	33.3	66.7	33.3	**100.0**
HumanEval	2	**100.0**	**100.0**	**100.0**	50.0	50.0	50.0	**100.0**	**100.0**

Table 8: Detailed breakdown of MixEval (Standard) sub-benchmark performance (% accuracy), with per-sub-benchmark sample counts (n). Methods are abbreviated: Std. PT (Standard Pre-Training), FI (FineInstructions), Nemo-CC (Nemotron-CC). We **bold** the strongest method within each corpus group.

		IPT Corpus (23B)			Nemotron-CC Corpus (300B)				
Sub-Benchmark	n	**Std. PT**	**IPT**	**FI**	**Std. PT**	**WRAP**	**Q&A**	**Nemo-CC**	**FI**
Overall		14.0	16.7	**19.2**	17.1	18.4	18.9	16.7	**21.8**
TriviaQA	331	1.9	5.8	**5.9**	9.4	9.4	**11.8**	11.0	9.8
MMLU	152	27.0	27.0	**27.6**	23.7	30.9	26.3	27.0	**33.6**
DROP	122	3.6	12.1	**29.2**	12.3	8.3	16.6	6.6	**35.3**
HellaSwag	61	**29.5**	26.2	24.6	26.2	**27.9**	**27.9**	18.0	24.6
CommonsenseQA	51	23.5	27.5	**29.4**	15.7	19.6	23.5	**35.3**	27.5
MMLU-Pro	90	11.1	8.9	**15.6**	11.1	11.1	11.1	10.0	**17.8**
BoolQ	32	**53.1**	46.9	43.8	50.0	53.1	53.1	**59.4**	46.9
AGIEval	65	15.8	**24.6**	**24.6**	21.1	**24.6**	19.3	12.3	19.3
BBH	20	5.0	**31.5**	15.0	**18.0**	12.5	12.5	14.0	9.0
PIQA	12	**58.3**	50.0	25.0	58.3	50.0	**75.0**	41.7	33.3
SIQA	26	**38.5**	30.8	34.6	26.9	**38.5**	23.1	26.9	26.9
ARC	6	0.0	**16.7**	**16.7**	**50.0**	**50.0**	**50.0**	16.7	**50.0**
OpenBookQA	8	25.0	12.5	**37.5**	37.5	**62.5**	12.5	25.0	37.5
GSM8k	7	0.0	**14.3**	**14.3**	0.0	0.0	0.0	0.0	**14.3**
MATH	12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GPQA	3	0.0	33.3	**66.7**	**33.3**	0.0	0.0	0.0	0.0
WinoGrande	2	**100.0**	0.0	50.0	**100.0**	50.0	50.0	0.0	0.0
MBPP	—	—	—	—	—	—	—	—	—
HumanEval	—	—	—	—	—	—	—	—	—

Table 9: Detailed breakdown of MixEval-Hard sub-benchmark performance (% accuracy), with per-sub-benchmark sample counts (n). Methods are abbreviated: Std. PT (Standard Pre-Training), FI (FineInstructions), Nemo-CC (Nemotron-CC). We **bold** the strongest method within each corpus group.