Stratified Selective Sampling for Instruction Tuning with Dedicated Scoring Strategy

Anonymous ACL submission

Abstract

Recent work shows that post-training datasets for LLMs can be substantially downsampled without noticeably deteriorating performance. However, data selection often incurs high computational costs or is limited to narrow domains. In this paper, we demonstrate that data selection can be both—efficient and universal—by using a multi-step pipeline in which we efficiently bin data points into groups, estimate quality using specialized models, and score difficulty with a robust, lightweight method. Task-based categorization allows us to control the composition of our final data—crucial for finetuning multi-purpose models. To guarantee diversity, we improve upon previous work using embedding models and a clustering algorithm. This integrated strategy enables high-performance fine-tuning with minimal overhead.

1 Introduction

Data selection has long been a central challenge in machine learning, aimed at curating datasets that maximise model performance. This has gained particular significance in the era of large language models (LLMs) (Albalak et al., 2024). LLM training typically spans three stages—pretraining, instruction-tuning, and alignment—with data selection playing a crucial role in each (e.g. Grattafiori et al., 2024b). However, the training objectives differ between each phase, and the goals of data selection also vary accordingly. While in pretraining the selection mechanisms are relatively coarse, relying on heuristics or lightweight classifiers to filter out the most undesirable data (Rae et al., 2021; Lee et al., 2022; Weber et al., 2024; Penedo et al., 2024), the post-training phase of instruction-tuning and subsequent alignment requires, in comparison, more rigorous selection strategies to guide the model toward desirable behaviors (Longpre et al., 2023; Conover et al., 2023; Wang et al., 2023; Grattafiori et al., 2024b).

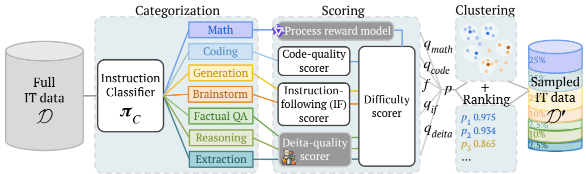


Figure 1: An illustration of our three-stage pipeline: we first classify instructions into seven types, then score each sample for instruction difficulty (f_i) and response quality (q_i). We rank their combined scores (p_i) and sample top scoring examples within embedding-space clusters as well as overall, while maintaining fixed category proportions.

Recent research on instruction tuning (IT) underscores the critical importance of data quality over sheer data quantity, establishing the benefits of IT data selection beyond merely reducing costs (see e.g. Zhou et al., 2023a; Liu et al., 2025; Qin et al., 2025), as larger datasets not only yield diminishing returns as they expand, but they can have detrimental effects on performance, especially with synthetic data (Ge et al., 2024; Diddee and Ippolito, 2025). On the other hand, given the abundance of available instruction data, one *has* to make a selection decision.

Manually curating effective instruction sets is both time-consuming and labour-intensive (Zhou et al., 2023a). A promising alternative is to automatically filter a small but diverse set of effective instructions from the extensive pool of available instruction data. Previous studies have proposed strategies to select "effective" instructions, focusing on three key dimensions of data assessment (Qin et al., 2025): *i)* **difficulty/relevance**, assessing whether a sample is trivial or challenging and how much it can therefore contribute to performance improvements, *ii)* **quality**, which refers to the usefulness and accuracy of responses, and *iii)* **diversity**, emphasising the scope of within-domain variability and cross-domain coverage of the data.

Prior work and its limitations. Prior work highlights the value of complex or challenging examples for post-training (Li et al., 2024a; Liu et al., 2024a), but often relies on prompting LLMs to estimate complexity (Liu et al., 2024b; Lu et al., 2024) and lacks domain generalizability (Muennighoff et al., 2025). Similarly, quality scoring methods (e.g., Liu et al., 2024b; Ge et al., 2024) often ignore task-specific scoring needs (e.g., solutions' correctness for *math* or *coding* problems), and some depend on large or proprietary models, raising cost and reproducibility issues (Chen et al., 2024). Domain-aware scoring strategies require a preprocessing step to bucket samples by task type. While some works mention this as part of data analysis (Ouyang et al., 2022a; Conover et al., 2023) or filtering pipelines (Grattafiori et al., 2024b), concrete methods and applications remain underexplored. Furthermore, as we detail in Section 2, only a few prior works consider all aspects of data assessment simultaneously (e.g., Liu et al., 2024b), and most evaluate on a single benchmark, limiting generalizability.

Our contributions. In this paper, we propose a robust instruction-tuning data selection framework that simultaneously considers difficulty, quality and diversity, and we demonstrate its effectiveness across a wide range of benchmarks. Our key contributions are as follows: *(i)* a novel data sampling strategy for instruction-tuning that integrates task types (e.g., *math*, *coding*, *generation*), difficulty, and quality scores, while preserving both inter- and intra-class diversity through clustering and ranking; *(ii)* an efficient method for classifying IT data samples by task type; *(iii)* a new and efficient approach to estimate difficulty scores using model performance across diverse benchmarks; *(iv)* task-specific quality scoring strategies, including custom-designed scorers, particularly for constrained generation and coding tasks focusing on responses' correctness; and *(v)* a large-scale evaluation spanning models of different families and sizes, with comparisons to strong baselines.

2 Related work

IT data selection. Prior work on IT data selection generally falls into two categories based on how they assess sample difficulty, quality, and diversity: *external scoring* and *model-inherent criteria*. **External scoring** methods rely on: *(i)* hand-crafted features such as coherence, grammaticality, naturalness and understandability (Cao et al., 2024); *(ii)* heuristics based on length or formatting (Zhao et al., 2024a; Muennighoff et al., 2025); and *(iii)* scores derived from LLMs of varying scales (*LLM-scorers*). Notable approaches leveraging LLM-scorers are: *AlpaGasus* (Chen et al., 2024), which prompts ChatGPT to score each data point; *InsTag* (Lu et al., 2024), which uses ChatGPT to generate open-ended tags for deriving complexity/diversity; *Deita* (Liu et al., 2024b), which fine-tunes LLaMA-1-13B on ChatGPT-annotated complexity/quality labels as *deita-scorers*; and *CaR* (Ge et al., 2024), which trains a 550M reward model to rank instruction-response pairs by quality. On the other hand, **model-inherent criteria** form another group of approaches that rely on signals from the target model itself, including: *LESS* (Xia et al., 2024), which estimates data influence using gradient information; *SelectIT* (Liu et al., 2024a), which measures uncertainty via token probability, prompt variation and multiple models' assessments; *SHED* (He et al., 2024), which clusters data and estimates impact using Shapley values; *instruction-following difficulty* (*IFD*, Li et al., 2024a) measuring discrepancies between the model's intrinsic generation capability and its desired response; and the approach by Li et al. (2024b), which evaluates sample utility based on how much it reduces loss when used as an in-context example.

Our approach combines both perspectives: we use domain-specific LLM-scorers to assess quality, and leverage various models' performances across benchmarks to estimate difficulty—that is, how likely an average model is to fail on a given instruction. For diversity, our approach closely follows *CaR* (Ge et al., 2024), which clusters data points and samples one representative from each cluster.

IT data categorization. Previous work has proposed various task types (e.g., *open QA*, *brainstorming*, *creative writing*), to guide IT data collection from human annotators (Ouyang et al., 2022a; Conover et al., 2023). Grattafiori et al. (2024b) fine-tuned Llama 3 8B for coarse-grained (e.g., *mathematical reasoning*) and fine-grained (e.g., *geometry and trigonometry*) topic classification to help filter low-quality samples, though they provide little detail on the methodology or intended downstream use. Other efforts impose tags or domain taxonomy on IT data to ensure diversity (Lu et al., 2024; Muennighoff et al., 2025). Dong et al. (2024) study how data composition across tasks affects model performance, but assume that ShareGPT only contains general alignment tasks—we found that nearly 30% of it actually consists of coding tasks. To our knowledge, no prior work leverages IT data categorization to apply task-specific scoring strategies during selection.

Utilities of IT data selection methods. Several papers criticized the effectiveness and cost-efficiency of existing IT data selection strategies. Zhao et al. (2024a) show that simply selecting the longest responses can outperform more complex methods while being significantly cheaper and easier to implement. Similarly, Diddee and Ippolito (2025) find that many sophisticated methods barely outperform random sampling under realistic conditions, and emphasize the cost-performance trade-off. However, most of these comparisons are limited to a single-source sampling setup, whereas a more practical scenario involves selecting from a pool of IT data sources. Additionally, prior evaluations focus on LLaMA models with a few exceptions using Mistral (Liu et al., 2025), leaving the generalizability of selection strategies across model families and sizes largely unexplored.

3 Methods

Our goal is to determine a subset of an arbitrarily large-scale instruction dataset that is effective when finetuning a given pre-trained base model. We define an effective dataset as one which achieves high performance on a given set of general evaluation tasks, while requiring relatively few model parameter updates. Formally, let $\mathcal{D} = \{d_i\}_{i=1}^{N}$ with $d_i = (x_i, y_i)$ be the full IT dataset of size N, where x_i represents an input sequence (i.e. an instruction) and y_i represents the corresponding output (i.e. desirable model response). We wish to select a subset $\mathcal{D}' \subset \mathcal{D}$ of size m (with $m \ll N$) that is most effective for instruction tuning.

Previous research has shown that *i)* **instruction difficulty** (see e.g. Li et al., 2024a; Liu et al., 2024a,b; Zhao et al., 2024b) *ii)* **response quality** (see e.g. Zhao et al., 2024a; Chen et al., 2024; Liu et al., 2024b) and *iii)* **diversity/composition** (see e.g. Ge et al., 2024; Lu et al., 2024) are crucial for effective data selection for LLM finetuning. We address these three aspects in a three-step pipeline as illustrated in Figure 1:

1. **Classification**. We train a lightweight classifier π_c to categorise all inputs x_i in $\mathcal{D}$ into one out of seven categories, which we denote as $l \in \mathcal{L}$.
2. **Scoring**. Each input–output pair (x_i, y_i) is scored using a category-specific quality scorer (yielding f_i) and a general-purpose difficulty scorer (yielding q_i); the results are combined into an overall preference score p_i.
3. **Clustering + Ranking**. Ultimately, we select the samples with the highest p_i while using a clustering approach to maintain diversity and minimise redundancies in $\mathcal{D}'$.

We detail each step in the remainder of this section.

3.1 Instruction Classification

Inspired by the use-case categories defined in Ouyang et al. (2022b), we establish the following task categories $\mathcal{L}$ for samples in instruction tuning datasets: *i)* **Math**, from simple calculation to problems requiring multi-step reasoning; *ii)* **Coding**, code generation tasks or programming-related question answering; *iii)* **Generation**, textual generation tasks including roleplaying, summarizing and rewriting passages; *iv)* **Reasoning**, questions requiring deductive/logical reasoning; *v)* **Brainstorming**, information-seeking and recommendation questions that require inductive reasoning, including classification tasks; *vi)* **Factual QA**, factual questions with simple facts as answers; and *vii)* **Extraction**, tasks requiring structured/answer extraction from textual contexts. We let two human annotators classify 80 samples from the popular MT-Bench dataset (Zheng et al., 2023a) and achieve high inter-annotator agreement (Cohen's Kappa = 0.8635), demonstrating the discriminability of our categories. We compare two different approaches to build a classification model: *i) LLM annotator* and *ii) SetFit classifier*.

With the LLM-annotator approach (Wei et al., 2022), we prompt instruction categorization by listing categories with brief explanations, followed by *"What is the category of the following task?"* (see Figure 12, Appendix A.1). Meanwhile, SetFit (Tunstall et al., 2022) is a few-shot learning method that tunes Sentence Transformers (Reimers and Gurevych, 2019) on labelled input pairs in a contrastive, Siamese manner. We manually identify approximately 250 samples strongly associated with each category (see Table 3, Appendix A.1) to train the SetFit classifier; hyperparameters are detailed in Appendix A.1.

For evaluation, we use the manually annotated MT-Bench dataset as the test set, assessing accuracy, macro F1-score, and Cohen's Kappa agreement with human judgment (see Table 1). While

Approach	LLM annotator/embedding model	Acc	F1	Cohen's Kappa
Zero-shot prompting	GPT-4o	**0.88**	**0.86**	**0.85**
	tiiuae/Falcon3-10B-Instruct	0.85	0.84	0.82
	meta-llama/Llama-3.1-8B-Instruct	0.76	0.65	0.71
SetFit classifier	NovaSearch/stella_en_400M_v5 †	0.85	0.81	0.82
	Lajavaness/bilingual-embedding-large	0.82	0.78	0.79
	NovaSearch/stella_en_1.5B_v5	0.66	0.60	0.60

Table 1: Evaluation results on instruction categorization. † denotes the chosen approach and model for our data selection pipeline.

zero-shot prompting with GPT-4o performs best, we choose the SetFit classifier for our pipeline due to its comparable performance and higher efficiency than larger LLMs like GPT-4o and Falcon3-10B-Instruct (Team, 2024b).

3.2 Scoring

Domain-agnostic difficulty scorer. Previous research suggests that *difficulty* (sometimes referred to as *complexity*) matters for data selection (see e.g. Liu et al., 2024b; Cao et al., 2024; Zhao et al., 2024b; Muennighoff et al., 2025), with more challenging data generally resulting in better model performance. However, existing difficulty metrics either often lack generality across domains (e.g. length of reasoning trace in response in Muennighoff et al., 2025) or are strongly influenced by spurious features (e.g. the widely used deita-complexity is strongly biased towards long sequences; see Figure 7 in Appendix A.2.1; Liu et al., 2024b).

Our goal is to train a general and robust difficulty scorer for our data selection pipeline that predicts how likely it is for an average model to solve a data point incorrectly, independent of its category l.

To source the training set $\mathcal{D}^{\text{diff}}$ for such a scorer, we collect 20k instruction-response pairs, evenly distributed across categories $l_i \in \mathcal{L}$ (data sources are listed in Table 4; proportions are shown in Figure 8). We evaluate every item with a heterogeneous pool of 18 instruction-tuned LLMs (see Table 5). Subsequently, we apply multiple preprocessing steps to the model scores: First, we normalise the item scores to the interval $[0, 1]$ and remove items with a score of 0 across all models, as they are likely to contain noise or annotation errors. To mitigate potential skews in the model performance distribution, we convert the absolute scores into *relative* deviations with respect to the model's mean performance on the corresponding category source dataset, by subtracting the average from the absolute score on each item. Ultimately, a difficulty target for every item is obtained by averaging over the model pool. We fine-tune a Qwen-3-8B backbone (Team, 2025) with a single-layer regression head to minimise the mean-squared error on this dataset (training details can be found in Appendix A.2.3).

Response quality scorer. Prior work shows that sample selection strategies based on response length or quality, as judged by external models, lead to better instruction tuning datasets (see e.g. Zhao et al., 2024a; Chen et al., 2024; Liu et al., 2024b). However, these strategies underestimate the diverse problem types within instruction tuning dataset, which may require different evaluation criteria. For example, the quality of responses to math and coding problems is heavily dependent on solution correctness, while constrained generation requires evaluation of adhered constraints. In this work, we designate a dedicated quality scorer for each category defined in Section 3.1, focusing on *i)* mathematical reasoning traces for *Math* (q_{math}), *ii)* code snippets for *Coding* (q_{code}), and *iii)* instruction-following capability for *Generation* and *Brainstorming* (q_{if}). For *Reasoning*, *Factual QA* and *Extraction*, we employ the *deita-quality scorer*[1] (q_{deita}, Liu et al., 2024b), a fine-tuned LLaMA-13B for quality assessment,

Process reward model. Process reward models (PRMs, Lightman et al., 2023; Uesato et al., 2022) are trained to verify steps in reasoning traces as they are common in mathematical reasoning. We score *Math* data points using Qwen2.5-Math-PRM-7B (Zhang et al., 2025b). We find double linebreaks and—if no double linebreaks present—single linebreaks as a delimiter to be a good heuristic to separate reasoning steps. As a reasoning trace breaks with a single erroneous step, we aggregate scores by taking the minimal score out of all steps within each trace, as the q_{math} score.

Code quality scorer. We design a quality scoring framework for *Coding* samples, drawing inspiration from Wadhwa et al. (2024). For each data point (x_i, y_i), we leverage code-oriented LLMs to: *(i)* assess the functional correctness of the code snippet in y_i with respect to the problem x_i, and *(ii)* produce a revised version that improves or fixes the original code (see Figure 13, Appendix A.3). The resulting score, q_{code}, is based on the *normalized Levenshtein similarity* between lines of the orig-

[1] https://huggingface.co/hkust-nlp/deita-quality-scorer

inal $(lo_0, ..., lo_n)$ and revised $(lr_0, ..., lr_m)$ code: $nls = (\max(n, m) - \text{lev}(lo, lr))/\max(n, m)$, where $\text{lev}(lo, lr)$ is the line-level Levenshtein distance. If the original code is functionally correct, we set $q_{\text{code}} = nls$; otherwise, $q_{\text{code}} = nls/2$. If no code snippet is present, we assign $q_{\text{code}} = 0.5$ if y_i is judged correct, and 0.0 otherwise.

To evaluate this scoring method, we use a 1K-sample test set from LiveCodeBench (Jain et al., 2024a), containing coding problems and LLM-generated responses.[2] Using Qwen/Qwen2.5-Coder-14B-Instruct as the reviewing model, our framework achieves 70% accuracy and a 0.412 Pearson correlation with binary correctness labels (see Appendix A.3 for details).

Instruction-following scorer. Taking inspiration from the IFEval benchmark (Zhou et al., 2023b) which defines "verifiable constraints" such as length (*"400 or more words"*) and keyword (*"without using the word sleep"*) constraints, we design a response quality scorer based on the fraction of expressed constraints ($\mathcal{C}_{exp}$) adhered by the response ($\mathcal{C}_{true}$). First, we use an LLM annotator to identify $\mathcal{C}_{exp}$, which comprises (span, constraint type) pairs $\{(s_i, c_i)\}_{i=1}^{n_{exp}}$, with s_i represents the textual span found and c_i is the corresponding constraint label. For example, given the prompt *"Write a funny blog post with 400 or more words about the benefits of sleeping in a hammock, without using the word sleep."*, $\mathcal{C}_{exp} = \{$(*400 or more words*, length), (*without using the word "sleep"*, keyword avoided), (*funny blog post*, writing type)$\}$. A list of considered constraint types is shown in Figure 14 (Appendix A.4). Next, $\mathcal{C}_{exp}$ is passed to a constraint checker module, which performs two steps:

1. Heuristic verification: We verify length, letter case, punctuation and keyword constraints, by adapting the IFEval verification script.
2. LLM-judge verification: We ask an LLM judge to assess constraints that cannot be verified heuristically (e.g., *"Does the following text follow the [writing type] constraint of [funny blog post]?"*).

This yields $\mathcal{C}_{true} = \{(s_j, c_j)\}_{j=1}^{n_{true}}$, with which we compute the quality score as $q_{\text{if}} = n_{true} * (n_{true}/n_{exp})$, giving more incentives to responses adhering to more constraints. If $\mathcal{C}_{exp}$ is empty, we ask an LLM judge to evaluate whether the response *i)* addresses the user's intent, while *ii)* respecting any constraints expressed in the prompt, and to provide a final *score* (1–10), which we use to compute the score as $q_{\text{if}} = score/10$.

Our analysis with the IFEval benchmark dataset containing sample responses from ten models as our test bed (see Table 8, Appendix A.4), shows that Qwen3-14B (Team, 2025) outperformed other medium-sized Instruct-LLMs as both LLM annotator and judge (see Table 9, Appendix A.4). It achieved a macro F1-score of 0.86 for identifying expressed constraints, a Pearson correlation coefficient of 0.523 at the instance-level, and 0.995 at the model-level, where it effectively replicated the IFEval model ranking.

Overall preference scores. For each $(x_i, y_i) \in \mathcal{D}$, we compute the preference score $p_i = f_i \cdot q_i$, where f_i and q_i are the difficulty and quality scores, respectively. Each of them is normalized using min-max scaling, with the 1st and 99th percentiles as the minimum and maximum values across all samples in $\mathcal{D}$. The quality scores are normalized per scorer (q_{math}, q_{code}, q_{if} and q_{deita}) as they have differing ranges. For multi-turn conversations, where each data point d_i consists of a sequence of turns $\{(x_0, y_0), ..., (x_T, y_T)\}$, we assign a category l_t to each turn and determine the conversation-level category l_i by either selecting the most frequent category or defaulting to the first one. We then compute turn-level difficulty and quality scores, f_t and q_t, based on their respective categories l_t. These are averaged across all turns to yield the overall conversation-level scores f_i and q_i.[3]

3.3 Sampling

Greedily choosing the highest-scoring samples often leads to redundancy in some domains and underrepresentation in others. Thus, maintaining diversity in the final dataset $\mathcal{D}'$ is essential. Since diversity is a property of the dataset as a whole—not of individual samples—selection should consider the dataset globally (e.g., Ge et al., 2024), rather than relying on iterative, sample-by-sample strategies (e.g., Liu et al., 2024b; Bukharin et al., 2024). Diversity can be promoted *top-down* by balancing category proportions (see e.g. Grattafiori et al., 2024b; Dong et al., 2024), or bottom-up by ensuring sufficient semantic dissimilarity among selected samples (e.g., Liu et al., 2024b; Ge et al., 2024; Lu et al., 2024).

[2] https://huggingface.co/spaces/livecodebench/code_generation_samples

[3] For a given conversation, quality scores are averaged separately for each category; q_i is then selected according to the main category l_i.

Dataset	#samples (*#turns*)
HuggingFaceH4/ifeval-like-data	5K
vicgalle/alpaca-gpt4	52K
nvidia/OpenMathInstruct-2[5]	52K
ai2-adapt-dev/flan_v2_converted	90K
openbmb/UltraInteract_sft (Coding)	115K
WizardLMTeam/WizardLM_evol_instruct_V2_196K	143K
theblackcat102/sharegpt-english	50K (392K)
microsoft/orca-agentinstruct-1M-v1[6]	200K (903K)
all	707K (1.75M)

Table 2: Instruction tuning dataset overview.

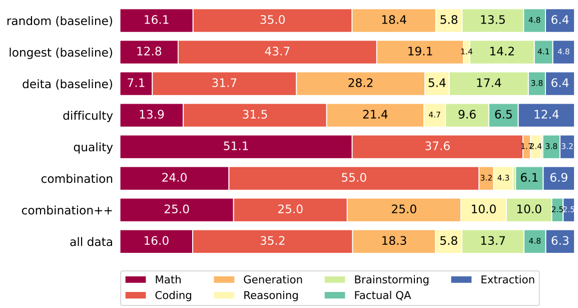


Figure 2: Category proportions of different sampling strategies for 100k samples.

As they are complementary, we propose a combination of both approaches: First, we determine the number of samples per category $l \in \mathcal{L}$, denoted m_l, ensuring balanced proportion of *Math*, *Coding*, *Generation* and others. Next, we embed all candidate samples within each category using a state-of-the-art sentence encoder (Reimers and Gurevych, 2019; Zhang et al., 2025a), and cluster them into J groups using k-means (Lloyd, 1982), with $J = m_l$. Let $\mathcal{K} = \{K_1, K_2, \ldots, K_J\}$ denote the resulting clusters. From each cluster K in $\mathcal{K}$, we select the sample with the highest preference score $p_{\max}(K) = \max_{i \in K} p_i$. To improve robustness to clusters with very low $p_{\max}(K)$ values, we discard clusters whose best sample falls below a predetermined threshold, which is set to be the γ^{th}-percentile of $\{q_i\}_{i=1}^{N_l}$ where N_l is the number of samples within the category l.[4] To reach the target of m_l samples per category, we select the highest-scoring samples from the remaining candidates within that category.

4 Experiments

4.1 Experimental Setup

Datasets. We use the IT datasets detailed in Table 2. In our experiments, the full IT dataset D is the aggregation of *all* listed datasets.

Models. While most experiment results use finetuned Mistral-7B-v0.3, we also demonstrate generalization across models of varying sizes and families: *i)* tiiuae/Falcon3-**10B**-Base, *ii)* meta-llama/Llama-3.1-**8B**, *iii)* mistralai/Mistral-**7B**-v0.3, *iv)* Qwen/Qwen2.5-**3B** and *v)* HuggingFaceTB/SmolLM2-**1.7B**.

Finetuning. We fine-tune each base model using the *SFTTrainer* from the Transformer Reinforcement Learning (TRL) library[7] (von Werra et al., 2020). Model-specific hyperparameters (e.g., learning rate) are selected for each model family and detailed in Appendix A.6. We also employ NEFTune (Jain et al., 2024b), a technique that improves the performance of chat models by injecting noise into embedding vectors during training.

Baselines. We compare against the following baseline methods for constructing $D' \subset D$: *(i) Random*, where we sample uniformly at random, *(ii) Longest* (Zhao et al., 2024a), where we include samples having the longest responses and *(iii) Deita* (Liu et al., 2024b), which ranks data points according to complexity and quality scores (o_{deita} and q_{deita}, resp.), and iteratively builds D' by adding samples that are dissimilar to those already selected.

Evaluation. We evaluate the success of our instruction tuning experiments by testing Models on a suite of benchmarks, covering a broad spectrum of model capabilities and giving a holistic picture of the tuning success. All tested benchmarks are listed in Table 11 in Appendix A.7.

4.2 Main Results

For every selection strategy we construct a subset $\mathcal{D}' = d_{i=1}^{m} \subset \mathcal{D}$ with size $m = 100,000$ and subsequently fine-tune each of the five base models on $\mathcal{D}'$. In addition to the baseline strategies described in Section 4.1, we investigate four principled variants that exploit the pipeline components introduced in Section 3.2:

i) Quality—sort $\mathcal{D}$ by q_i and select the top m items;

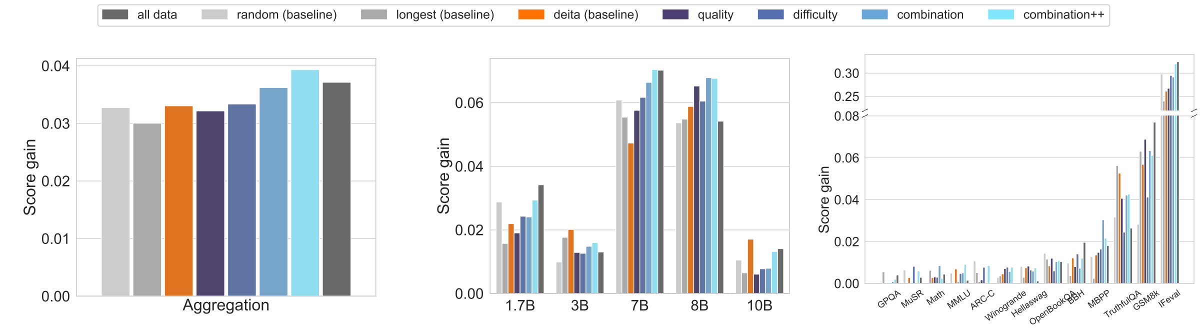
ii) Difficulty—sort by f_i and select the top m items;

[4] γ is a hyperparameter and set to 75.

[5] Without augmented problems.

[6] Randomly sampled.

[7] https://huggingface.co/docs/trl/sft_trainer



(a) Aggregation models & benchmarks (b) Aggregation benchmarks (c) Aggregation models

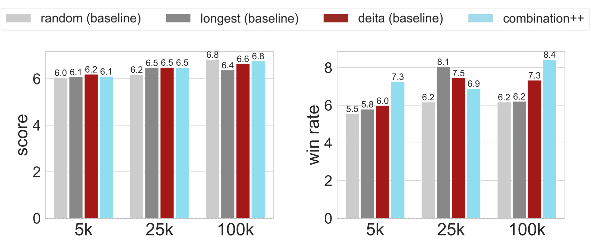
Figure 3: Performance gains over the base model for different sampling strategies with 100k samples. We aggregate the results (a) across all 5 tested models and all 13 benchmarks; (b) across all benchmarks separated by models; (c) across all models separated by benchmarks.

iii) Combination—sort by the preference score $p_i = f_i \cdot q_i$ and select the top m items;

iv) Combination++—as in *combination*, but with sampling via the clustering-and-quota procedure from Section 3.3.

Figure 2 shows task-type distributions across sampling strategies, including baselines. Figure 3a reports average performance gains over the untuned base model for all evaluation benchmarks. The full pipeline (*combination++*) achieves the highest overall performance, surpassing every baseline and even the model trained on the entire source set with $|\mathcal{D}| > 707$k.

To examine robustness, we break the results down by model size (Figure 3b). *Combination++* yields the most consistent gains of all tested conditions for all five bases and is outperformed by full-data training in only two cases, confirming that the proposed sampling approach transfers well across models. A benchmark-wise analysis (Figure 3c) shows similarly stable improvements, with *combination++* delivering consistent performance throughout.



(a) MT-Bench (b) Alpaca Eval 2.0

Figure 4: Performance on LLM-as-a-judge benchmarks.

For completeness, we also evaluate our approach on benchmarks using LLM-as-a-judge. Due to the high cost of the proprietary LLM-judges, we limit ourselves to evaluating models trained on Mistral-7B-Base under a single experimental condition. As shown in Figure 4, for $m = 100,000$, *combination++* performs best in the length-controlled AlpacaEval 2.0 (Dubois et al., 2024), and matches random sampling on MT-bench (Zheng et al., 2023b).

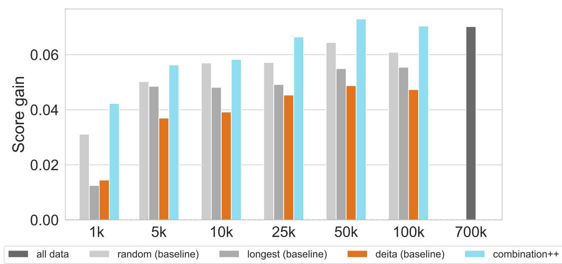
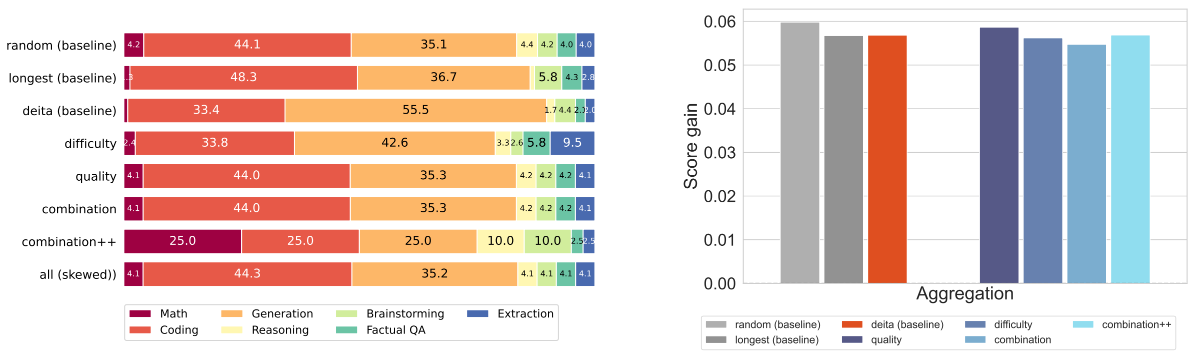


Figure 5: Average benchmark score gains over the base model, when finetuning Mistral-7B-Base with different dataset sizes sampling using various strategies.

Robust downscaling We next test whether the full pipeline's benefits persist when the target size m is varied. Fixing the base model to Mistral-7B-Base, we sample additional datasets $\mathcal{D}'$ of 1k, 10k, 25k, 50k items with *combination++* and all baselines. Figure 5 demonstrates that our method outperforms the alternatives across every scale: notably, with only $m = 50,000$ (7% of the source data), it surpasses the performance resulting from full-data training. For benchmark-specific plots, see Appendix A.8.1.

We, again, evaluate on LLM-as-a-judge benchmarks under limited setups. As before, *combination++* performs comparably to other strategies on MT-bench, while outperforming them on AlpacaEval for $m = 5,000$ and on average performance for



(a)

(b)

Figure 6: (a) Category proportions of $\mathcal{D}_{\text{skewed}}$ and the 25k subsets sampled with various selection strategies; (b) Average benchmark scores, when finetuning Mistral-7B-Base with data sampled from the skewed distribution using various strategies.

$m = 25,000$.

The surprising insensitivtiy of MT-Bench to sampling strategies may stem from the consistent presence of GPT-4-generated responses in every sampled datasets (see Figure 10, Appendix A.5), which is known to strongly influence MT-Bench outcomes (Panickssery et al., 2024; Stureborg et al., 2024; Wataoka et al., 2024).

Robustness to skewed data distributions We now probe how selection strategies perform when the candidate pool $\mathcal{D}$ is itself highly imbalanced. To simulate this, we create a *skewed* source set $\mathcal{D}_{\text{skewed}} \subset \mathcal{D}$ by randomly selecting two categories $l_1, l_2 \in \mathcal{L}$ and retaining *only* items with labels $l_i \in l_1, l_2$, along with a small (4%) residue from all other categories. This yields a strongly biased data distribution of size $N_{\text{skewed}} = 366\text{k}$ (see Figure 6a bottom, for the resulting category distribution).

From $\mathcal{D}_{\text{skewed}}$, we sample $m = 25\text{k}$ items with the same strategies as previously and fine-tune Mistral-7B-Base on each $\mathcal{D}'_{\text{skewed}}$. We expected diversity-aware sampling strategies like *Deita* and *combination++* to mitigate bias by enforcing either semantic spread and explicit quotas. However, none of the strategies show a clear advantage, with *random* sampling performing with a slight, insignificant edge. On closer inspection of the category distributions of the sampled data in Figure 6a, we find that sampling strategies without explicitly encouraging diversity, such as *quality* and *difficulty*, mostly maintain the bias of $\mathcal{D}_{\text{skewed}}$, while *Deita* exacerbates it. *Combination++* successfully debiases the sample; however, this balancing does not translate into improved model performance.

5 Discussion and Conclusion

Effective data selection for instruction tuning is increasingly important as we handle the increasing amount of available data of mixed quality. Nevertheless, the results of data selection pipelines have often been brittle, struggling to generalise across training setups (Diddee and Ippolito, 2025; Zhao et al., 2024a).

In this paper, we address this challenge by explicitly covering the whole spectrum of possible instruction domains and tailoring scoring strategies that are apt to judge the utility of samples in those domains. We show the robustness of our approach by testing it across diverse settings (model families, scales, and sample sizes) and evaluating it on a wide range of common benchmarks. Our experiments provide further evidence for the necessity of data selection, as our sampling not only significantly reduces the amount of data required for training, but also outperforms setups trained on the full source data. Despite its more elaborate design—including multiple scoring methods—our pipeline remains computationally efficient due to targeted scoring.

While our pipeline outperforms baselines in almost all settings, it shows no significant gains when applied to a strongly skewed source distribution. We caution against over-interpreting this outcome, as we only evaluate on a single type of distributional bias (toward two of seven categories). A more complete robustness assessment would require testing across various bias configurations, which is computationally infeasible and thus left for future work.

Limitations

Difficulty scorer. A major issue in collecting data for our difficulty scorer is a certain unreliability in the evaluation of model responses. Previous research shows that evaluations oftentimes have weak robustness to e.g. the prompt formatting (Sclar et al., 2024; Weber et al., 2023a,b; Polo et al., 2024), bias of LLMs-as-a-judge (Panickssery et al., 2024; Stureborg et al., 2024; Wataoka et al., 2024) or errors in post-processing (such as issues in extracting the answer from the model response). For example, GPT4o showed overall weaker performance on some of our evaluated subsets than some small open-source models. Upon closer inspection, we encountered that – while providing the correct answer – GPT4o generally tends not to follow the formatting of the given few-shot examples and rather responds in an open-form manner, resulting in failing the tight response search masks of the used evaluation frameworks. While we try to mitigate this issue as much as possible, we cannot guarantee that difficulty scores exactly reflect a model's capacity to solve a given data point.

Quality scorer. We did not develop nor have dedicated quality scorers for samples belonging to *Reasoning*, *Factual QA* and *Extraction* categories. However, we hypothesis that process reward models (PRMs) could be adapted to *Reasoning* tasks beyond math, such as spatial (e.g., Wu et al., 2024) and deductive reasoning (e.g., Seals and Shalin, 2024), given appropriate training data. *Factuality* assessment is a long-standing research problem that has become more relevant in the era of LLMs. Wei et al. (2024) introduced *SAFE (Search-Augmented Factuality Evaluator)*, an LLM-agent-based method for automatically assessing long-form factuality in model response. Although significantly cheaper than human annotators (up to 20×), SAFE still incurs costs of \$20–\$40 per 100 prompt-response pairs. For *Extraction* tasks, future work might draw inspiration from recent advances in RAG evaluation (Yu et al., 2025). These directions, however, are beyond the scope of this paper.

Robustness to skewed data. Our results indicate that we cannot get performance gain in setups with very skewed data. However, the experimental setup is very limited, testing only skew in a single variation and more experiments are needed to draw conclusions. Further, there are many other skews that can occur in practice that we did not evaluate our sampling pipeline against.

References

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. 2024. A survey on data selection for language models. *Transactions on Machine Learning Research*. Survey Certification.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. 2022. A framework for the evaluation of code generation models. `https://github.com/bigcode-project/bigcode-evaluation-harness`.

Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. 2024. Data diversity matters for robust instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3411–3425, Miami, Florida, USA. Association for Computational Linguistics.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2024. Instruction mining: Instruction data selection for tuning large language models. In *First Conference on Language Modeling*.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Harshita Diddee and Daphne Ippolito. 2025. Chasing random: Instruction selection strategies fail to generalize. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1943–1957, Albuquerque, New Mexico. Association for Computational Linguistics.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *Preprint*, arXiv:1511.06931.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 177–198, Bangkok, Thailand. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Mahong Xia, Zhang Li, Boxing Chen, Hao Yang, Bei Li, Tong Xiao, and JingBo Zhu. 2024. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 464–478, Miami, Florida, USA. Association for Computational Linguistics.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. 2025. Generating structured outputs from language models: Benchmark and studies. *Preprint*, arXiv:2501.10868.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024a. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Aaron Grattafiori and 1 others. 2024b. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Yexiao He, Ziyao Wang, Zheyu Shen, Guoheng Sun, Yucong Dai, Yongkai Wu, Hongyi Wang, and Ang Li. 2024. SHED: Shapley-based automated dataset refinement for instruction fine-tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021a. Measuring coding challenge competence with apps. *NeurIPS*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024a. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *arXiv preprint*. ArXiv:2403.07974 [cs].

Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024b. NEFTune: Noisy embeddings improve instruction finetuning. In *The Twelfth International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, arXiv:1705.03551.

Sakaguchi Keisuke, Le Bras Ronan, Bhagavatula Chandra, and Choi Yejin. 2019. Winogrande: An adversarial winograd schema challenge at scale.

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, and 1 others. 2023. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36:47669–47681.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. Mlqa: Evaluating cross-lingual extractive question answering. *arXiv preprint arXiv:1910.07475*, arXiv: 1910.07475.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024a. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7602–7635, Mexico City, Mexico. Association for Computational Linguistics.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024b. One-shot learning as instruction data prospector for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4586–4601, Bangkok, Thailand. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Liangxin Liu, Xuebo Liu, Derek Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024a. Selectit: Selective instruction tuning for llms via uncertainty-aware self-reflection. *Advances in Neural Information Processing Systems*, 37:97800–97825.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024b. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Ziche Liu, Rui Ke, Yajiao Liu, Feng Jiang, and Haizhou Li. 2025. Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6595–6611, Albuquerque, New Mexico. Association for Computational Linguistics.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

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

Mistral AI. 2025. Mistral small 3. `https://mistral.ai/news/mistral-small-3`. Apache 2.0 License.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2023. Learning deductive reasoning from synthetic corpus based on formal logic.

In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25254–25274. PMLR.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022a. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022b. Training language models to follow instructions with human feedback. *arXiv preprint*. ArXiv:2203.02155 [cs].

Arjun Panickssery, Samuel Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802.

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

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. 2024. Efficient multi-prompt evaluation of LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yulei Qin, Yuncheng Yang, Pengcheng Guo, Gang Li, Hang Shao, Yuchen Shi, Zihan Xu, Yun Gu, Ke Li, and Xing Sun. 2025. Unleashing the power of data tsunami: A comprehensive survey on data assessment and selection for instruction tuning of language models. *Transactions on Machine Learning Research*. Survey Certification.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, and 1 others. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

S Seals and Valerie Shalin. 2024. Evaluating the deductive competence of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8614–8630, Mexico City, Mexico. Association for Computational Linguistics.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *Proceedings of the International Conference on Learning Representations*. International Conference on Learning Representations.

Rickard Stureborg, Dimitris Alikaniotis, and Yoshi Suhara. 2024. Large language models are inconsistent and biased evaluators. *arXiv preprint arXiv:2405.01724*.

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. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. 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)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Qwen Team. 2024a. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwen3.

TII Team. 2024b. The falcon 3 family of open models.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient Few-Shot Learning Without Prompts. *arXiv preprint*. ArXiv:2209.11055 [cs].

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

David Vilares and Carlos Gómez-Rodríguez. 2019. HEAD-QA: A healthcare dataset for complex reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 960–966, Florence, Italy. Association for Computational Linguistics.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Nalin Wadhwa, Jui Pradhan, Atharv Sonwane, Surya Prakash Sahu, Nagarajan Natarajan, Aditya Kanade, Suresh Parthasarathy, and Sriram Rajamani. 2024. CORE: Resolving Code Quality Issues using LLMs. In *FSE*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and 1 others. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. 2024. Self-preference bias in llm-as-a-judge. *arXiv preprint arXiv:2410.21819*.

Lucas Weber, Elia Bruni, and Dieuwke Hupkes. 2023a. The icl consistency test. *arXiv preprint arXiv:2312.04945*.

Lucas Weber, Elia Bruni, and Dieuwke Hupkes. 2023b. Mind the instructions: a holistic evaluation of consistency and interactions in prompt-based learning. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 294–313.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 116462–116492. Curran Associates, Inc.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. *arXiv preprint*. ArXiv:2109.01652 [cs].

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Zixia Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V Le. 2024. Long-form factuality in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. 2024. Mind's eye of LLMs: Visualization-of-thought elicits spatial reasoning in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *2018 IEEE/ACM 15th international conference on mining software repositories (MSR)*, pages 476–486. IEEE.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2025. *Evaluation of Retrieval-Augmented Generation: A Survey*, page 102–120. Springer Nature Singapore.

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

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025a. Jasper and stella: distillation of sota embedding models. *Preprint*, arXiv:2412.19048.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024a. Long is more for alignment: a simple but tough-to-beat baseline for instruction fine-tuning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Minghao Li, Fei Huang, Nevin L Zhang, and Yongbin Li. 2024b. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16776–16789.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023a. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

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. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

A Appendix

A.1 Instruction Categorization – details

We present the zero-shot prompt for classifying instructions in Figure 12. As for training the SetFit classifier, we leverage the training data detailed in Table 3, and we employ the following hyperparameters. Note that training with SetFit consists of two phases behind the scenes: finetuning embeddings and training a differentiable classification head. As a result, some of the training arguments can be tuples, where the two values are used for each of the two phases, respectively.

- `batch_size`=(16, 2)
- `num_epochs`=(1, 15)
- `end_to_end`=True (train the entire model end-to-end during the classifier training phase)
- `body_learning_rate`=(2e-5, 1e-5), the second value is the learning rate of the Sentence Transformer body during the classifier training phase
- `head_learning_rate`=1e-4
- `max_steps`=500

A.2 Difficulty Scorer – details

A.2.1 Correlations of difficulty/complexity metrics with input length

As the length of an instruction might in some cases reflect the difficulty of the datapoint, there is no causal relationship between the two. A scorer that is trained to predict difficulty should therefore not rely on input length as a feature to rely its prediction upon. We show in Figure 7 how deita complexity is strongly correlated with input length, while the difficulty scorer we present in this paper is not. While this is not conclusive evidence that the complexity scorer is relying on input length as a feature in its prediction, these results are an indicator that it might. Further investigation is required to determine whether this relationship is causal.

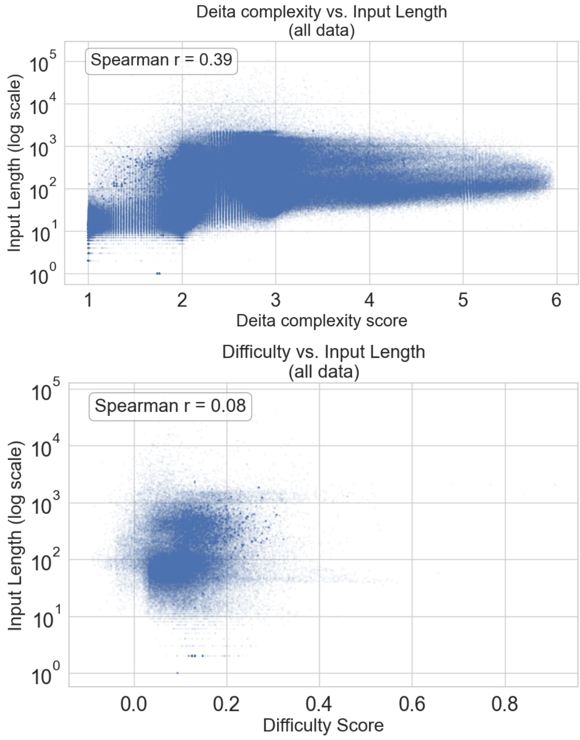


Figure 7: Relation between complexity/difficulty scores and length of the input sequence. The length of the input sequence should, in most cases, be unrelated to the difficulty of resolving it. Any correlation is therefore spurious.

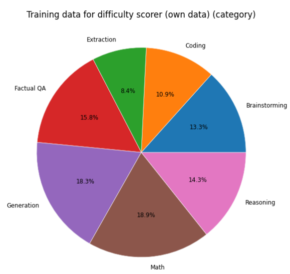


Figure 8: Setfit proportions of training data for difficulty scorer

A.2.2 Data generation difficulty scorer – details

Figure 8 show the proportions of different instruction-response pairs that we use a training data for our difficulty scorer. The subsequent Table 4 shows the respective data source from which we obtained the data, as well as the type of evaluation metric that we use to evaluate the 18 LLMs from Table 5.

During the data collection for the difficulty scorer, we collect training sets from different benchmarks as well as data points from OpenAssistent (Köpf et al., 2023) for examples of open-ended generation. We evaluate these samples using the existing evaluation frameworks LM-evaluation harness (Gao et al., 2024), big-code evaluation harness (Ben Allal et al., 2022) and FastChat (Zheng et al., 2023b). For the LLM-as-a-judge evaluation, we use GPT4o as a judge and evaluate only a subset of 4 LLMs from Table 5 that we deem representative in terms of capabilities (gpt-4o-2024-08-06, Mistral-Small-24B-Instruct-2501, Mistral-7B-Instruct-v0.3 and SmolLM2-1.7B-Instruct).

Category	**Training data** (*subset*)	**#Samples**
Math	nvidia/OpenMathInstruct-2, AI-MO/NuminaMath-CoT	250
Coding	openbmb/UltraInteract_sft (*Coding*), microsoft/orca-agentinstruct-1M-v1 (*code*), HuggingFaceH4/no_robots (*Coding*), lissadesu/codeqa_v3	253
Generation	HuggingFaceH4/no_robots (*Generation, Rewrite, Summarize*), HuggingFaceH4/ifeval-like-data, declare-lab/InstructEvalImpact (*Creative, Professional*), iamketan25/roleplay-instructions-dataset	250
Extraction	HuggingFaceH4/no_robots (*Closed QA, Extract*)	250
Factual QA	HuggingFaceH4/no_robots (*Open QA*), basicv8vc/SimpleQA	255
Brainstorming	declare-lab/InstructEvalImpact (*Informative, Argumentative*), HuggingFaceH4/no_robots (*Brainstorming, Classify*), matt-seb-ho/WikiWhy	250
Reasoning	renma/ProofWriter, hitachi-nlp/ruletaker, lucasmccabe/logiqa, lucasmccabe/logiqa, tasksource/strategy-qa	250

Table 3: Training data overview for SetFit classifier.

Dataset	**Subset**	**Split**	n **samples**	**Eval. metric**
GSM8K (Cobbe et al., 2021)	Default	train	1192	exact match
Math (Hendrycks et al., 2021c)	Algebra	train	322	exact match
	Counting & Probability	train	264	exact match
	Geometry	train	220	exact match
	Intermediate Algebra	train	233	exact match
	Number Theory	train	314	exact match
	Prealgebra	train	348	exact match
	Precalculus	train	238	exact match
IFEval-like	Default	-	1990	instance level loose acc
MBPP (Austin et al., 2021)	Default	train & val	448	pass@1
OpenBookQA (Mihaylov et al., 2018)	Default	train	299	acc
ARC (Clark et al., 2018)	Challenge	train	464	acc
bAbI (Dodge et al., 2016)	Default	train	224	exact match
CommonsenseQA (Talmor et al., 2019)	Default	train	248	acc
CoQA (Reddy et al., 2019)	Default	train	380	F1
DROP (Dua et al., 2019)	Default	train	383	F1
FLD (Morishita et al., 2023)	Default	train	739	exact match
	Logical Formula Default	train	750	exact match
HeadQA (Vilares and Gómez-Rodríguez, 2019)	English	train	689	acc
	Spanish	train	703	acc
JSONSchemaBench (Geng et al., 2025)	Easy	train	200	schema compliance & json validity
	Medium	train	198	schema compliance & json validity
	Hard	train	179	schema compliance & json validity
LogiQA (Liu et al., 2020)	LogiEval	train	490	exact match
	LogiQA2	train	416	acc
MLQA (Lewis et al., 2019)	49 lang. combinations	val	715	F1
TriviaQA (Joshi et al., 2017)	Default	train	1389	exact match
OpenAssistent (Köpf et al., 2023)	Default	train	5318	LLM-as-a-judge
APPS (Hendrycks et al., 2021a)	introductory	train	422	pass@1
	interview	train	303	pass@1
	competition	train	289	pass@1
CONALA (Yin et al., 2018)	Default	train	97	Bleu

Table 4: Datasets used in difficulty scorer training.

A.2.3 Training the difficulty scorer – details

We equip regular CausalLLMs with a regression head by pooling the final dense layers and adding a linear projection to a scalar output, and finetune these models on the difficulty scoring problem. During finetuning, we use the hyperparameters detailed in Table 6. We finetune four different base models as difficulty scorer (Llama-3.1-8B (Grattafiori et al., 2024a), Qwen-2.5-7B (Team, 2024a), Qwen3-4B and Qwen3-8B (Team, 2025)) and converge on Qwen3-8B as it produces the best performance on our i.i.d. evaluation data.

A.3 Code-quality Scorer – details

The prompt used by LLM-annotator/judge for deriving code-quality scores can be found in Figure 13. Table 7 presents the evaluation results of various LLMs as the code reviewer.

A.4 Instruction-following Scorer – details

Prompts used by LLM-annotator/judge for deriving instruction-following scores can be found in Figure

Model	Type	Size
SmolLM2-135M-Instruct (Allal et al., 2025)	Univ.	135M
SmolLM2-360M-Instruct (Allal et al., 2025)	Univ.	360M
Qwen2.5-0.5B-Instruct (Team, 2024a)	Univ.	0.5B
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)	Math	1.5B
Qwen2.5-Coder-1.5B-Instruct (Hui et al., 2024)	Code	1.5B
Qwen2.5-1.5B-Instruct (Team, 2024a)	Univ.	1.5B
SmolLM2-1.7B-Instruct (Allal et al., 2025)	Univ.	1.7B
Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	Math	7B
Qwen2.5-Coder-7B-Instruct (Hui et al., 2024)	Code	7B
Qwen2.5-7B-Instruct (Team, 2024a)	Univ.	7B
Mistral-7B-Instruct-v0.3 (Jiang et al., 2023)	Univ.	7B
Qwen2.5-14B-Instruct (Team, 2024a)	Univ.	14B
Mistral-Small-24B-Instruct-2501 (Mistral AI, 2025)	Univ.	24B
Qwen2.5-32B-Instruct (Team, 2024a)	Univ.	32B
Qwen2.5-Coder-32B-Instruct (Hui et al., 2024)	Code	32B
Qwen3-32B (Team, 2025)	Univ.	32B
gpt-4o-mini-2024-07-18 (Hurst et al., 2024)	Univ.	unk
gpt-4o-2024-08-06 (Hurst et al., 2024)	Univ.	unk

Table 5: Models that were evaluated to obtain difficulty scores for our difficulty scorer training set.

Hyperparameter	Value
batch_size	1
gradient_accumulation	16
learning_rate	1e-5
lr_scheduler_type	linear
num_train_epochs	8
warmup_steps	100
max_seq_length	2048
weight_decay	0.01
neftune_noise_alpha	10

Table 6: Hyperparameter details for training the difficulty scorer

LLM reviewer	acc	Pearson's r
deepseek-ai/DeepSeek-Coder-V2-Lite-Instruct	0.64	0.278
deepseek-ai/DeepSeek-R1-Distill-Qwen-14B	0.66	0.389
Qwen/Qwen2.5-Coder-14B-Instruct †	**0.70**	**0.412**
Qwen/Qwen3-14B *(non-coding LLM)*	**0.70**	0.411

Table 7: Evaluation results of various LLMs as code reviewer (acc), and Pearson correlation coefficient (r) of resulting code quality scores with LiveCodeBench benchmarks binary correctness labels. † denotes the chosen LLM annotator/judge for our instruction-following scorer.

14, 15 and 16. As our test dataset, we collected responses from 10 LLMs on the IFEval benchmark (see Table 8), available on open-llm-leaderboard's dataset collection of evaluation details.[8] Table 9 presents the evaluation results of various LLMs as the annotator/judge.

[8] https://huggingface.co/open-llm-leaderboard

Model	IFEval score
meta-llama/Llama-3.3-70B-Instruct	0.90
Qwen/Qwen2.5-14B-Instruct-1M	0.84
allenai/Llama-3.1-Tulu-3-70B-SFT	0.81
tiiuae/Falcon3-7B-Instruct	0.76
ibm-granite/granite-3.1-8b-instruct	0.72
microsoft/Phi-3-medium-128k-instruct	0.60
abacusai/Smaug-34B-v0.1	0.50
Qwen/Qwen2.5-32B	0.41
google/gemma-1.1-2b-it	0.31
databricks/dolly-v2-7b	0.20

Table 8: Performance of 10 considered LLMs on the IFEval benchmark.

LLM annotator/judge	macro F1	Pearson's r instance-level	Pearson's r model-level
Qwen/Qwen2.5-7B-Instruct	0.80	0.503	0.986
meta-llama/Llama-3.1-8B-Instruct	0.83	0.454	0.982
tiiuae/Falcon3-10B-Instruct	0.84	0.515	0.969
Qwen/Qwen3-14B †	**0.86**	**0.523**	**0.995**

Table 9: Evaluation results of various LLMs as annotator/judge on identifying expressed constraints (macro-F1), and Pearson correlations coefficient (r) of resulting instruction-following scores with IFEval benchmarks scores at both instance-level and model-level. † denotes the chosen LLM annotator/judge for our instruction-following scorer.

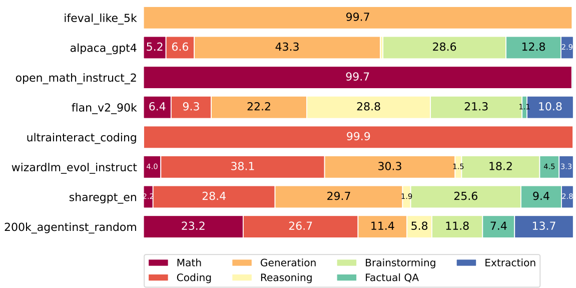


Figure 9: Category proportions of IT datasets used in the experiments.

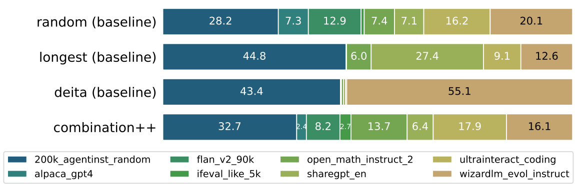


Figure 10: Source dataset proportions of different sampling strategies for 100k samples.

A.5 Datasets – details

Figure 9 shows SetFit classification results for IT datasets used in the experiments. Whereas Figure 10 shows the composition of source datasets for various sampling strategies. Deita is basically

Hyperparameter	falcon-10B	llama-8B	mistral-7B	qwen-3B	smollm-1.7B
batch_size	4	8	8	8	8
gradient_accumulation	16	8	8	8	8
learning_rate	2.0e-05	1.0e-05	5.0e-06	5.0e-06	5.0e-06
lr_scheduler_type			cosine		
num_train_epochs			2		
attn_implementation			flash_attention_2		
warmup_ratio			0.03		
max_seq_length			2048		
weight_decay			0.1		
neftune_noise_alpha			5		
use_liger			True		

Table 10: Hyperparameter details for finetuning.

Benchmark	Category ($\mathcal{L}$)	Evaluation details
IFeval (Zhou et al., 2023b)	Generation	0-shot; prompt level strict acc
GPQA (Rein et al., 2024)	Multiple	0-shot
BBH (Suzgun et al., 2022)	Reasoning	3-shot; multiple-choice
MuSR (Sprague et al., 2024)	Reasoning	0-shot; multiple-choice
Math (Hendrycks et al., 2021c)	Math	4-shot
ARC-C (Clark et al., 2018)	Multiple	0-shot; multiple-choice
GSM8k (Cobbe et al., 2021)	Math	5-shot
Hellaswag (Zellers et al., 2019)	Multiple	0-shot; multiple-choice
MMLU (Hendrycks et al., 2021b)	Factual QA	0-shot; multiple-choice
TruthfulQA (Lin et al., 2021)	Factual QA	0-shot; multiple-choice
Winogrande (Keisuke et al., 2019)	Multiple	0-shot; multiple-choice
MBPP (Austin et al., 2021)	Coding	3-shot; pass@1
OpenBookQA (Mihaylov et al., 2018)	Extraction	0-shot; multiple-choice

Table 11: List of used benchmarks with associated category and evaluation details.

dominated by only two datasets: microsoft/orca-agentinstruct-1M-v1 (200k sampled) and WizardLMTeam/WizardLM_evol_instruct_V2_196K.

A.6 Finetuning – details

We finetune all models in our experiments on one node of 4 x NVIDIA H100-SXM5 Tensor Core-GPUs (94 GB HBM2e). Table 10 details the hyperparameters used for finetuning.

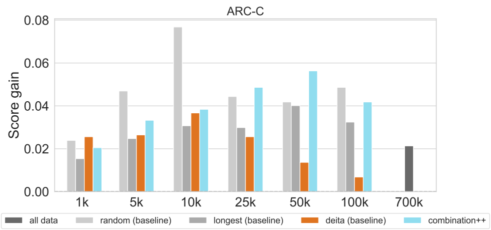
A.7 Evaluation – details

In the following, we list the benchmarks and the respective evaluation details, such as few-shot settings and other information about formatting and response extraction.

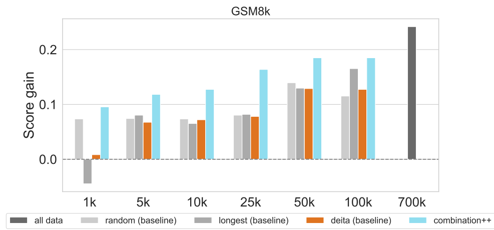
A.8 Results – details

A.8.1 Results – scaling by benchmark

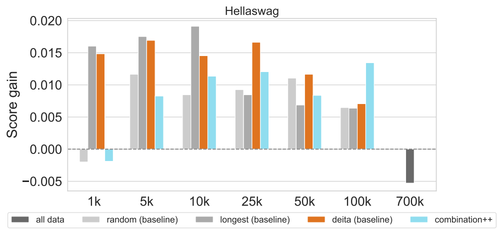
We present here the results of our scaling experiments, without aggregating them across evaluation benchmarks.



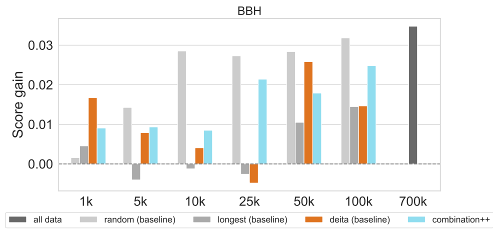
(a) ARC-challenge (Clark et al., 2018)



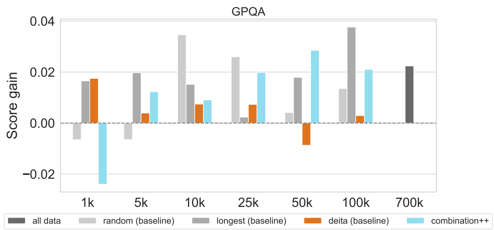
(b) GSM8k (Cobbe et al., 2021)



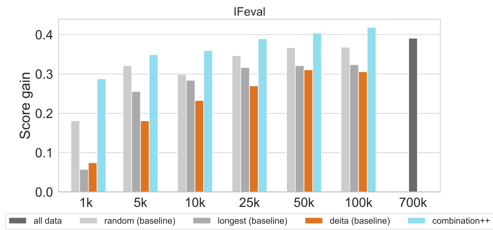
(c) HellaSwag (Zellers et al., 2019)



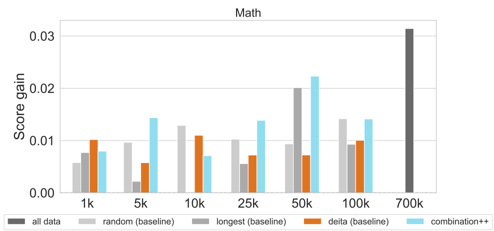
(d) BBH (Suzgun et al., 2022)



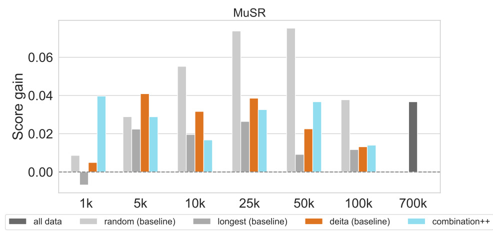
(e) GPQA (Rein et al., 2024)



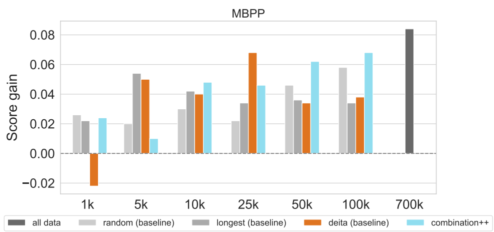
(f) IFeval (Zhou et al., 2023b)



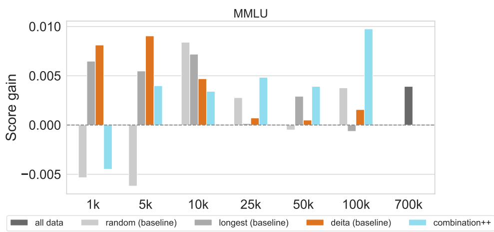
(g) Math (Hendrycks et al., 2021c)



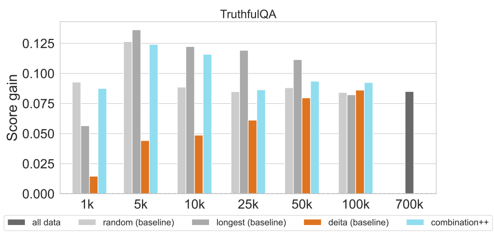
(h) MuSR (Sprague et al., 2024)



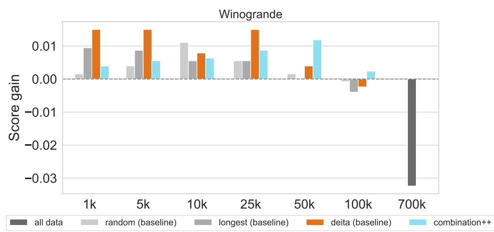
(i) MBPP (Austin et al., 2021)



(j) MMLU (Hendrycks et al., 2021b)



(k) TruthfulQA (Lin et al., 2021)



(l) Winogrande (Keisuke et al., 2019)

Figure 11: Results scaling across all benchmark datasets

```
Given the following categories:
- Math (math questions and math reasoning problems)
- Coding (programming tasks or coding questions)
- Generation (creative generation tasks with constraints, including roleplaying)
- Reasoning (logical deductive reasoning tasks that are neither math nor coding)
- Brainstorming (information-seeking or recommendation questions requiring explanation, or classification tasks)
- Factual QA (simple factual questions, without any context)
- Extraction (extraction tasks, including QA, from a given textual passage)

What is the category of the following task? Please respond only in JSON format (e.g., {{"answer": "Generation"}})

### Task ###
{input}
```

Figure 12: Zero-shot prompt for instruction categorization.

```
### Task ###
1. Given the following user's PROMPT and system's RESPONSE, please review the code snippet in the RESPONSE.
2. Focusing on functional correctness, give the final verdict: 'correct' vs 'incorrect'.
3. Extract the original code snippet, write "no code" if there's no code snippet.
4. If the original code is correct, simply write "no revision", otherwise propose a code revision to improve the code.
5. Provide your answer in JSON format, with "review", "final_verdict", "code_original" and "code_revision" as keys.

### User's PROMPT ###
{instruction}

### System's RESPONSE ###
{output}
```

Figure 13: Zero-shot prompt for code review and code revision.

```
### Task ###
1. Given a USER's prompt, decide whether the constraints from the list below are expressed in the USER's prompt (yes/no).
2. Provide the expressed constraints in JSON format with the expressed constraint as the key
   and the constraint type as the value if the respective constraint is expressed.

### List of Constraints ###
- letter_case, e.g., lowercase, all capitals
- placeholder_and_postscript
- repeat_prompt, e.g., repeat the request
- output_combination, e.g., multiple responses, separate the response
- choose_output, e.g., choose answer from given options
- output_format, e.g., json format, markdown format, bulleted list, formatted title, highlighted sections
- keyword_included, e.g., included words
- keyword_avoided, e.g., avoided words
- keyword_frequency, e.g., five hashtags, 'but' two times, letter 'r' at least 3 times
- language, e.g., english, two languages
- length, e.g., number of words, number of sentences, number of paragraphs
- punctuation, e.g., no commas, quotation
- start_and_ending, e.g., start with 'Hello', end with 'Thank you!'
- writing_style (e.g., shakespeare, easy-to-read, 5-year-old, persuasive)
- writing_type (e.g., letter, email, proposal, poem)
- topic (e.g., love)

### Examples ###
{few-shot_examples}

### Question ###
USER: {instruction}
```

Figure 14: Few-shot prompt for constraint identification.

```
### Task ###
Answer the following questions. Provide the answer in JSON format with the question number as the key and the answer as
the value (true/false).
Questions:
{questions}

ASSISTANT:
{output}
```

Figure 15: Zero-shot prompt for constraint verification.

```
### Task ###
Given the USER's prompt and ASSISTANT's response below, analyze whether the response addresses USER's intents properly,
while respecting any constraints expressed in the prompt. Based on these judgments, provide the final score of response
quality in the range of 1 to 10, in JSON format ('score' as key and quality score as value).

USER: {instruction}
ASSISTANT:
{output}
```

Figure 16: Zero-shot prompt for response evaluation.