LASER: Stratified Selective Sampling for Instruction Tuning with Dedicated Scoring Strategy

Paramita Mirza[1]*, **Lucas Weber**[2]*, **Fabian Küch**[2]

[1]ScaDS.AI & TU Dresden, Germany,
[2]Fraunhofer IIS, Germany

paramita.mirza@tu-dresden.de {lucas.weber, fabian.kuech}@iis.fraunhofer.de

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

Large language models (LLMs) can perform a wide range of text-based tasks through chat interfaces. Their generalist abilities stem from an extensive post-training phase, during which they are optimized to generate useful responses to user queries (Sanh et al., 2022; Mishra et al., 2022; Wei et al., 2022; Ouyang et al., 2022). The choice of training data in this phase has a major impact on model performance (e.g. Zhou et al., 2023a; Xia et al., 2024; Liu et al., 2024b; Chen et al., 2024; Ge et al., 2024).

Prior work identifies three key properties of effective instruction-tuning (IT) data (Zhou et al., 2023a): *i)* **difficulty/relevance**, reflecting how much a query contributes to learning (Li et al., 2024b; Liu et al., 2024a,b; Zhao et al., 2024b); *ii)* **quality**, the usefulness and accuracy of responses (Zhao et al., 2024a; Chen et al., 2024; Liu et al., 2024b); and *iii)* **diversity**, the scope of within-domain variability and cross-domain coverage of the data (Ge et al., 2024; Lu et al., 2024). Although many approaches address one or more of these aspects, they often face limitations that make them difficult to apply in practice.

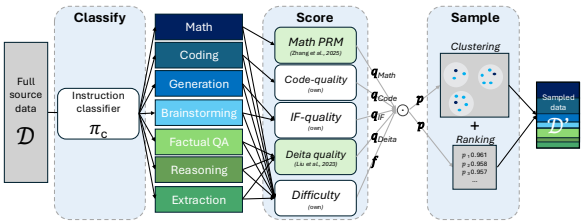


Figure 1: Overview of our three-stage pipeline, LASER. Instructions are first classified into seven types, then routed and scored for difficulty (f) and quality (q). Their combined scores (p) are ranked, and top examples are sampled both within embedding clusters and overall, while preserving fixed category proportions.

Most existing methods cover only a subset of criteria (e.g., *diversity* in Chen et al. 2024; *quality* and *diversity* in Ge et al. 2024). To our knowledge, DEITA (Liu et al., 2024b) is the only **complete** pipeline that integrates all three simultaneously. Beyond completeness, prior work can be further refined in terms of **generality** and **robustness**. Many heuristic-based methods cannot be applied to different tasks (Muennighoff et al., 2025) or domains (Zhou et al., 2023a). Broader approaches—that do not limit themselves explicitly to a single domain—are usually evaluated only on narrow domains or task-types (e.g., open-ended language generation), fine-tuned on a small set of models (typically from the Llama or Mistral family with 3B–8B parameters), or limited to a single source dataset. Outside these settings, they often prove brittle, failing to consistently outperform random sampling or simple heuristics such as a *longest* baseline (Diddee and Ippolito, 2025; Zhao et al., 2024a). Ultimately, many methods are **expensive** as they rely on large or closed-weight LLM judges (Chen et al., 2024; Lu et al., 2024).

*These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 19949–19974
November 4-9, 2025

We address these limitations with LASER (Label-Aware Scoring and clustERing), a robust, complete, and efficient data selection pipeline.[1]

Our **key contributions** are the following:

(i) A novel, *complete* and *robust* data sampling strategy for instruction-tuning that integrates *efficient* scoring of difficulty and quality via routing while maintaining both inter- and intra-class diversity through stratified sampling and clustering;

(ii) A new and general approach to estimate the difficulty of instructions;

(iii) Task-specific quality scoring strategies, including custom-designed scorers, particularly for constrained generation and coding tasks focusing on responses' correctness; and

(iv) A large-scale evaluation spanning diverse model families and sizes, on a wide range of benchmarks covering various task types, 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)* handcrafted 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., 2024b), measuring discrepancies between the model's intrinsic generation capability and its desired response; and the approach by Li et al. (2024c), which evaluates sample utility based on how much it reduces loss when used as an in-context example.

Our approach embraces external and model-inherent perspectives simultaneously: 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 categorisation. 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., 2022; Conover et al., 2023). Grattafiori et al. (2024) 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, while our analysis shows that nearly 30% of it actually consists of coding tasks. To our knowledge, no prior work leverages IT data categorisation 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 lim-

[1] Code, model access, and replication details will be released upon publication at https://github.com/Fraunhofer-IIS/LASER

ited 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 instruction source 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 large and general set of 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 source dataset of size N, where x_i represents an input sequence (i.e. an instruction) and y_i represents the corresponding output (i.e. the 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., 2024b; 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 the three-step LASER 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 general-purpose difficulty scorer (yielding f_i) and a category-specific quality 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. (2022), 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., 2023) 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*.

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.

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 19, 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 12, 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 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

3.2.1 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 lack generality across domains (e.g. length of reasoning trace in response in Muennighoff et al., 2025, or the DEITA *complexity scorer*[2] as shown in Figure 10b) or are strongly influenced by spurious features (e.g. the widely used DEITA-complexity is strongly biased towards long sequences; see Figure 11; Liu et al., 2024b).

Our goal is to train a general and robust difficulty scorer that predicts how likely it is for an average model to solve a data point incorrectly, independent of its category l. We denote this difficulty score as f.

To source the training set $\mathcal{D}_{\text{diff}}$ for such a scorer, we collect 20k instruction-response pairs, approximately equally distributed across categories $l_i \in \mathcal{L}$. We list the data sources in Table 13 and show the category proportions in Figure 8. Then, we evaluate every item with a heterogeneous pool of 18 instruction-tuned LLMs (see Table 4). We continue by applying 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 and will not yield any valuable learning signal. We then go on to convert the absolute scores into *relative* deviations with respect to the model's mean performance on the corresponding dataset, by subtracting the average from the absolute score on each item (illustrated in Figure 9). This step has the effect of weighing wrong responses of strong models more than wrong responses of weaker models (and vice versa). Further, it mitigates potential skews in the model performance distribution (e.g. if many models perform weakly on a specific dataset). Ultimately, we obtain difficulty targets f by averaging over the model pool for every item.

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.2.2).

Difficulty scorer evaluation. We evaluate the effectiveness and the validity of the difficulty scoring. For effectiveness, we sample 25k data points from $\mathcal{D}$ (as it is described in Section 4.1) either at random or following the difficulty scores. We then finetune Mistral-7B-v0.3 (Jiang et al., 2023) on the resulting datasets. Appendix A.2.3 shows how the difficulty scorer helps improve performance on benchmarks independently of their domain. For validity (i.e. whether we are really measuring some notion of difficulty), we score the CodeForces section of DeepMind's code-contests dataset (Li et al., 2022) and find high correlations with its human-annotated difficulty scores (details in Appendix A.2.4).

3.2.2 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 data, 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 $l \in \mathcal{L}$ defined in Section 3.1.

DEITA quality scorer. For *Reasoning*, *Factual QA* and *Extraction* samples, we employ the DEITA *quality scorer*[3] (Liu et al., 2024b), a finetuned LLaMA-13B for quality assessment, which yields the q_{deita} score.

Process reward model. For *Math* data, we assess the soundness of mathematical reasoning using a specialised process reward model (Qwen2.5-Math-PRM-7B; Zhang et al., 2025). Process reward models (PRMs; Lightman et al., 2024; Uesato et al., 2022) are trained to verify steps in reasoning traces as they are common in mathematical reasoning. 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.

[2] hkust-nlp/deita-complexity-scorer

[3] hkust-nlp/deita-quality-scorer

Code quality scorer. We design a quality scoring framework for *Coding* samples by 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 20, Appendix A.3.2). The resulting score, q_{code}, is based on the *normalized Levenshtein similarity* between lines of the original $(lo_0, ..., lo_n)$ 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.[4] 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.3.2 for details).

Instruction-following scorer. For *Generation* and *Brainstorming* data, we implement an if-quality scorer. Influenced by 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 21 (Appendix A.3.1). 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 6, Appendix A.3.1), shows that Qwen3-14B (Team, 2025) outperformed other medium-sized Instruct-LLMs as both LLM annotator and judge (see Table 7, Appendix A.3.1). 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.

Quality scorer evaluation. Similar to the difficulty scorer, we demonstrate the effectiveness of our task-specific quality scorers for ranking-based data selection, particularly when compared to the general-purpose DEITA quality scorer and random sampling. The corresponding results are presented in Appendix A.3.3.

3.3 Sampling

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. f_i and q_i are normalized per scorer as they have differing ranges, using min-max scaling, with the 1st and 99th percentiles as the minimum and maximum values across all samples in $\mathcal{D}$. For multi-turn conversations, where each data point d_i consists of a sequence of turns $\{(x_0, y_0), ..., (x_T, y_T)\}$, these scores are averaged across all turns to yield the overall conversation-level scores f_i and q_i.

Clustering + Ranking. Greedily choosing the highest-scoring samples often leads to redundancy in some domains and under-representation 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—

[4] livecodebench/code_generation_samples

Dataset	#samples (*#turns*)
HuggingFaceH4/ifeval-like-data	5K
vicgalle/alpaca-gpt4 (Taori et al., 2023)	52K
nvidia/OpenMathInstruct-2 *(w/o augmented problems)* (Toshniwal et al., 2024)	52K
ai2-adapt-dev/flan_v2_converted (Longpre et al., 2023)	90K
openbmb/UltraInteract_sft (Coding) (Yuan et al., 2024)	115K
WizardLMTeam/WizardLM_evol_instruct_V2_196K (Xu et al., 2024)	143K
theblackcat102/sharegpt-english	50K (392K)
microsoft/orca-agentinstruct-1M-v1 *(random subset)* (Mitra et al., 2024)	200K (903K)
all	707K (1.75M)

Table 2: Source dataset $\mathcal{D}$

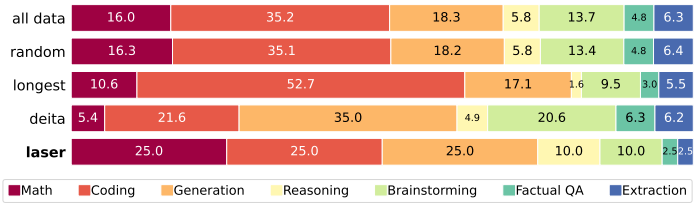


Figure 2: Category distribution in IT datasets with varied sampling.

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., 2024; 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).

As they are complementary, we propose a combination of both approaches: First, we determine the number of samples per category $l \in \mathcal{L}$ via a fixed quota, denoted m_l, ensuring a balanced proportion of *Math*, *Coding*, *Generation* and so on. Next, we embed all candidate samples within each category using a state-of-the-art sentence encoder (Reimers and Gurevych, 2019; Zhang et al., 2024), 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[5] of $\{p_i\}_{i=1}^{N_l}$ where N_l is the number of samples within the category l. To reach the target of m_l samples per category, we select the highest-scoring samples from the remaining candidates within that category.

Given the large value of m_l and the high-dimensional nature of the input embeddings, we employ *MiniBatchKMeans* from *scikit-learn* with the `k-means++` initialization method and a batch size of 2048.

Sampling evaluation. We evaluate the effectiveness of our stratified sampling and clustering as we did for the difficulty and quality scorer evaluation in Sections 3.2.1 and 3.2.2. In addition, we demonstrate the efficiency and robustness of LASER's sampling technique, particularly in comparison with DEITA (Liu et al., 2024b) and CAR (Ge et al., 2024). The respective results are reported in Appendix A.4.1.

4 Experiments

4.1 Experimental Setup

Datasets. We use the aggregation of *all* listed datasets in Table 2 as the source dataset $\mathcal{D}$ in all experiments, if not specified otherwise.

Models. While in most experiments we finetune Mistral-7B-v0.3, we also demonstrate generalisation 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[6] (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.

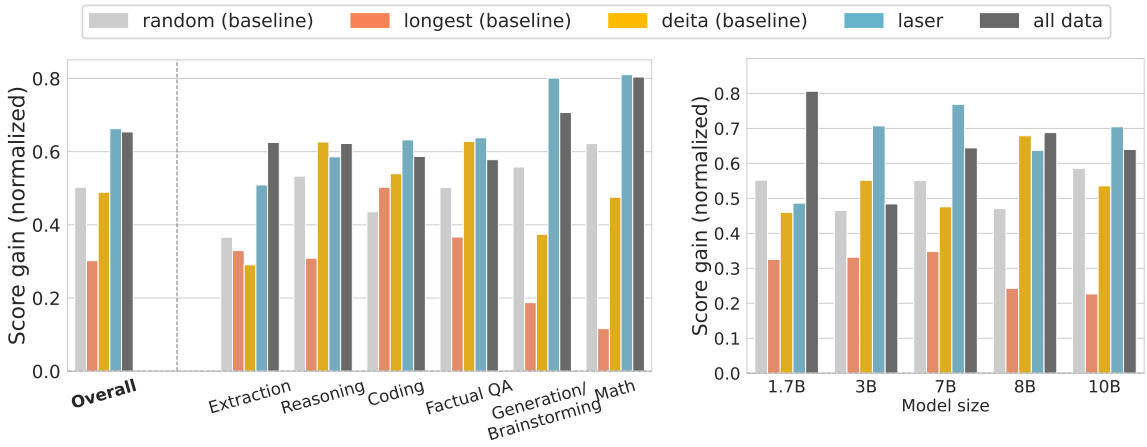
Baselines. We compare against the following alternative methods for constructing $D' \subset D$:

i) random—sampling uniformly at random;
ii) longest—including samples having the longest responses;[7]
iii) deita (Liu et al., 2024b)—ranking data points according to complexity and quality scores (o_{deita} and q_{deita}, resp.), and iteratively builds D' by adding *dissimilar* samples based on embedding distances.

[5] γ is a hyperparameter and set to 80.

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

[7] In the case of multi-turn samples, we only consider the last turn's responses.



(a) Score gains per benchmark category

(b) Score gains per model size

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

Category ($\mathcal{L}$)	Benchmark	Evaluation details	lm_eval
Math	$\text{GSM8k}_{\text{cot}}$ (Cobbe et al., 2021)	8-shot; chain-of-thought	gsm8k_cot_llama
	$\text{GSM8k}_{\text{cot-0-shot}}$ (Cobbe et al., 2021)	0-shot; chain-of-thought	gsm8k_cot_zeroshot
	Math (Hendrycks et al., 2021c)	4-shot	leaderboard_math_hard
Coding	HumanEval (Chen et al., 2021)	0-shot; pass@1	humaneval_instruct
	MBPP (Austin et al., 2021)	3-shot; pass@1	mbpp_instruct
Generation/ Brainstorming	AlpacaEval (Dubois et al., 2024)	length-controlled win-rate	-
	IFeval (Zhou et al., 2023b)	0-shot; prompt-level strict-acc	leaderboard_ifeval
Reasoning	ARC-C (Clark et al., 2018)	0-shot; multiple-choice	arc_challenge
	BBH (Suzgun et al., 2023)	3-shot; multiple-choice	leaderboard_bbh
	GPQA (Rein et al., 2024)	0-shot	leaderboard_gpqa
	Hellaswag (Zellers et al., 2019)	0-shot; multiple-choice	hellaswag
	MuSR (Sprague et al., 2024)	0-shot; multiple-choice	leaderboard_musr
	Winogrande (Sakaguchi et al., 2020)	0-shot; multiple-choice	winogrande
Factual QA	AGIEval (Zhong et al., 2024)	0-shot; multiple-choice	agieval_nous
	MMLU (Hendrycks et al., 2021b)	0-shot; multiple-choice	mmlu
	TruthfulQA (Lin et al., 2022)	0-shot; multiple-choice	truthfulqa_mc2
Extraction	OpenBookQA (Mihaylov et al., 2018)	0-shot; multiple-choice	openbookqa

Table 3: Evaluation benchmarks with associated category and evaluation details.

Evaluation. We evaluate the instruction-tuned models on a suite of benchmarks listed in Table 3, leveraging popular evaluation frameworks *LM-evaluation harness* (Gao et al., 2024) and *AlpacaEval*[8] (Dubois et al., 2024). We report *normalized score gain* as the main evaluation metric, computed as the performance improvement over base models, scaled with min-max normalization.

4.2 Results

4.2.1 Main results

For each selection strategy, we construct a subset $\mathcal{D}' = d_{i=1}^{m} \subset \mathcal{D}$ of size $m = 100,000$, and subsequently fine-tune each of the five base models on $\mathcal{D}'$. Figure 2 shows the resulting category distributions across sampling strategies, including the baselines. Figure 3a reports normalized score gains over the untuned base models for all evaluation benchmarks and models. LASER achieves the highest overall performance, surpassing all baselines and even the models trained on the entire source set ($|\mathcal{D}| > 707$k). We examine the robustness by breaking the aggregated results down by model size (Figure 3b). LASER yields the most consistent gains of all tested conditions for all five bases and outperforms the random sampling in all but the smallest model, confirming that the proposed sampling approach transfers well across models.

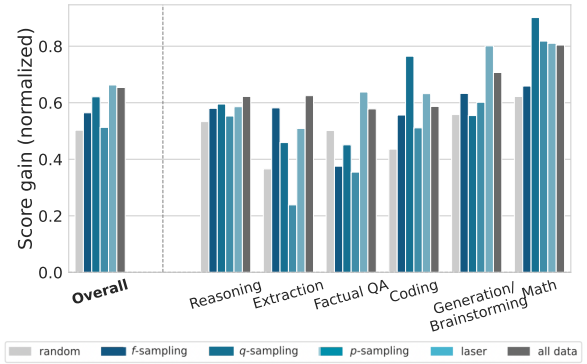


Figure 4: Average score gains over the base model when ablating LASER with 100k samples: with only f, q and p as ranking criteria.

To better understand the contribution of each LASER component, we ablate sampling by ranking with only difficulty scores (f), quality scores (q), or their combination (p)—without clustering or stratification—and compare these variants to the full pipeline. As shown in Figure 4, using f or q alone yields clear improvements over random sampling. However, their full potential is realized only when combined within the complete LASER pipeline. Notably, relying solely on p for

[8]We use `openai/gpt-oss-120b` as the LLM-judge.

ranking provides only marginal gains, and even underperforms f- or q-based sampling individually, highlighting the importance of diversifying the sampling through stratification and clustering in LASER.

Hence, all three properties (difficulty, quality and diversity) are necessary but not sufficient for sampling data that consistently performs well. This is intuitive: highly challenging queries answered incorrectly provide little benefit, while accurate responses to trivial questions yield weak training signals. Another salient result is the high performance of q-sampling for *Coding* and *Math*. While the soundness of responses (i.e. q-scores) is arguably more important in these domains, we also find that q- and f-scores in our source data $\mathcal{D}$ are generally negatively correlated, with especially high negative correlations in Coding ($r = -.38$) and Math ($r = -.43$). Shifting emphasis onto the difficulty (f-scores and p-scores) might therefore reduce the response quality in these domains and explain the performance gaps observed in Figure 4. We find similar relationships between quality and difficulty for the source datasets $\mathcal{D}_{weak}$ and $\mathcal{D}_{strong}$ in the following section.

4.2.2 Additional robustness experiments

We continue to demonstrate the robustness of the full LASER pipeline for different target sample sizes m and variations in the source data distribution $\mathcal{D}$, such as differences in source data quality and skewed data distributions.

Downscaling m. As different users may require instruction data sets of different sizes, a selection pipeline should improve sampling at every target scale. To verify this, we fix the base model to Mistral-7B-Base and constructs datasets $\mathcal{D}'$ of {1k, 5k, 10k, 25k, 50k, 100k} items using LASER, as well as *random* and *all data* as baselines. Figure 5 demonstrates that our method outperforms the alternatives across scales, except when $m = 1,000$. Notably, with only $m = 100,000$ (14% of the source data), it surpasses the performance of full-data fine-tuning. We provide benchmark-specific plots in Figure 24 (Appendix A.7), reported without normalizing *score gain*.

Effectiveness shifts in distribution of $\mathcal{D}$. Next, it is very probable that the average effectiveness of the source data $\mathcal{D}$ changes from scenario to scenario. We therefore test LASER with two shifted source distributions: $\mathcal{D}_{\text{strong}}$ consisting of the full Tülu v3 (Lambert et al., 2024)—a meticulously optimised dataset, which can be considered highly effective—and $\mathcal{D}_{\text{weak}}$ composed of multiple datasets that have previously proved to be less effective for instruction tuning (see Appendix A.5.2 for details).

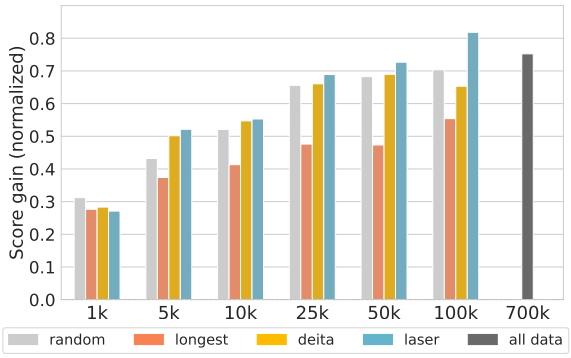


Figure 5: Average score gains over the base model, when finetuning Mistral-7B-Base using instruction datasets with varying sample sizes.

We sample $m \in \{1k, 5k, 10k, 25k, 50k, 100k\}$ data points from both sources $\mathcal{D}_{\text{strong}}$ and $\mathcal{D}_{\text{weak}}$ using LASER and random sampling. The results of finetuning Mistral-7B-v0.3 are shown in Figure 6. Generally, LASER performs robustly on different source datasets, with fewer fluctuations when changing the sampling size and comparably more improvements for the stronger than for the weaker source dataset. Interestingly, the performance gains using LASER over random are more pronounced for $\mathcal{D}_{\text{weak}}$ when sampling smaller target datasets, potentially distilling the few better examples from the weaker source.

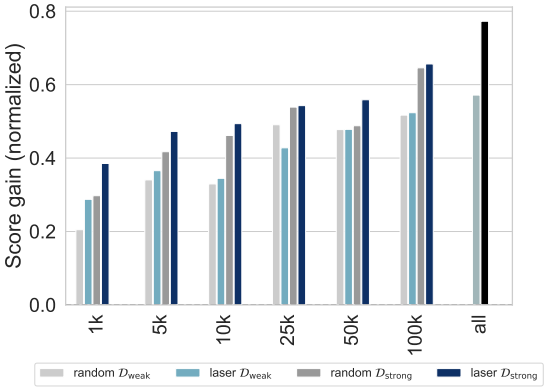
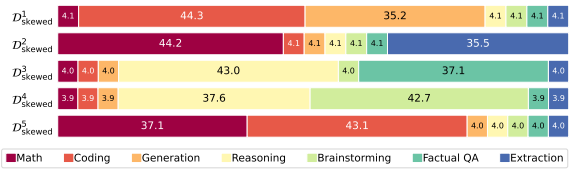
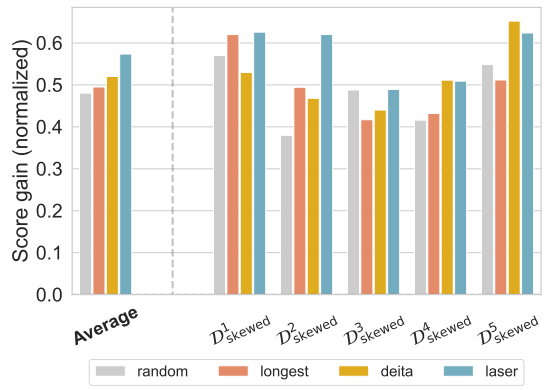


Figure 6: Performance on source datasets of different effectiveness ($\mathcal{D}_{\text{weak}}$ and $\mathcal{D}_{\text{strong}}$) for different sample sizes. For comparison, "all" shows performance when using full $\mathcal{D}_{\text{weak}}$ or $\mathcal{D}_{\text{strong}}$ without sampling.



(a)



(b)

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

Skewed distribution of $\mathcal{D}$. Ultimately, $\mathcal{D}$ might be very homogeneous (i.e. contains mostly a specific data type). To simulate this, we create five *skewed* source sets $\mathcal{D}_{\text{skewed}}$ by randomly selecting two categories $l_1, l_2 \in \mathcal{L}$ for each $\mathcal{D}_{\text{skewed}}$ and retaining only items with labels $l_i \in l_1, l_2$, along with a small (4%) residue from all other categories, yielding a strongly biased data distribution. We show the proportions of each category for all five new source datasets in Figure 7a.

From each $\mathcal{D}_{\text{skewed}}$, we sample 25k items with the same strategies as previously and fine-tune Mistral-7B-Base on each of the resulting $\mathcal{D}'_{\text{skewed}}$. We expected diversity-aware sampling strategies like DEITA and LASER to mitigate bias by enforcing either semantic spread and explicit quotas. Indeed, the diversity-aware sampling strategies on average outperform the naive methods, with LASER having an edge over DEITA, in terms of performance improvements as illustrated in Figure 7b. Upon closer inspection, we find that only LASER reduces the introduced bias of categories whereas DEITA exacerbates it (see Appendix A.5.2 for details). Next to that, we observe a key limitation of DEITA's sampling method in this setup: only adding dissimilar examples leads to quick saturation when source data is homogeneous and, thereby, hitting a ceiling of possible samples that can be sampled with DEITA.

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, various shifts in the source data distribution) 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 only little improvements when sampling large portions from weak source data (as shown in Section 4.2.2). This may reflect a potential ceiling effect on possible performance given the weak source material, suggesting that improving the data itself—rather than relying solely on elaborate sampling techniques—might be necessary. In such cases, iterative data refinement seems promising, especially when guided by reliable scorers for assessing sample difficulty and quality. The modular setup of LASER allows for further incremental improvements of different scorers in 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 (Weber et al., 2023a,b; Sclar et al., 2024; Polo et al., 2024b), 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 hypothesize 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.

Performance with weak source data. Our results indicate modest performance gains in setups with very weak source data. However, there might be a ceiling for possible performance in such settings, where data selection alone might not be sufficient to achieve good performance. In such cases, iterative data refinement appears to be a promising approach, particularly when supported by reliable scorers for assessing sample difficulty and quality.

Acknowledgments

We thank Rishiraj Saha Roy, Lucas Druart and Christian Kroos for their valuable feedback on earlier versions of this manuscript. This work was supported in part by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the OpenGPT-X project (no. 68GX21007D) and in part by the Free State of Bavaria in the DSgenAI project (Grant Nr.: RMF-SG20-3410-2-18-4). We also gratefully acknowledge the Centre for Information Services and High Performance Computing [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] and ScaDS.AI at the Technical University of Dresden for providing its facilities for experimental work.

References

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.

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*, abs/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.

Justyna Brzezińska. 2020. Item response theory models in the measurement theory. *Communications in Statistics-Simulation and Computation*, 49(12):3299–3313.

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.

Li Cai, Kilchan Choi, Mark Hansen, and Lauren Harrell. 2016. Item response theory. *Annual Review of Statistics and Its Application*, 3(1):297–321.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

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 preprint*, abs/1803.05457.

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*, abs/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 H. Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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 *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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. 2024. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*.

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.

Aaron Grattafiori and 1 others. 2024. The Llama 3 Herd of Models.

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 *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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*, abs/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*, abs/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.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. 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)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2023. Longform: Effective instruction tuning with reverse instructions. *Preprint*, arXiv:2304.08460.

John P. Lalor, Hao Wu, and Hong Yu. 2016. Building an evaluation scale using item response theory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 648–657, Austin, Texas. Association for Computational Linguistics.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *ArXiv preprint*, abs/2411.15124.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024a. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. 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.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, and 7 others. 2022. Competition-level code generation with alphacode. *ArXiv preprint*, abs/2203.07814.

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. 2024c. 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. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

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. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024a. Selectit: Selective instruction tuning for llms via uncertainty-aware self-reflection. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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 *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

FM Lord. 1968. Statistical theories of mental test scores. *(No Title)*.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

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

Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andres Codas, Yadong Lu, Wei-ge Chen, Olga Vrousgos, Corby Rosset, Fillipe Silva, Hamed Khanpour, Yash Lara, and Ahmed Awadallah. 2024. AgentInstruct: Toward Generative Teaching with Agentic Flows.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2023. Learning deductive reasoning from synthetic corpus based on formal logic. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, 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*, abs/2501.19393.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. 2024. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024a. tinybenchmarks: evaluating llms with fewer examples. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail

Yurochkin. 2024b. Efficient multi-prompt evaluation of llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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*.

Pedro Rodriguez, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. 2021. Evaluation examples are not equally informative: How should that change NLP leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4486–4503, Online. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, 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, and 21 others. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *ArXiv preprint*, abs/2404.02823.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

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.

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

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

Qwen Team. 2025. Qwen3.

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

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *ArXiv preprint*, abs/2410.01560.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient Few-Shot Learning Without Prompts.

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*, abs/2211.14275.

Wim J Van der Linden. 2018. *Handbook of item response theory: Three volume set*. CRC Press.

Clara Vania, Phu Mon Htut, William Huang, Dhara Mungra, Richard Yuanzhe Pang, Jason Phang, Haokun Liu, Kyunghyun Cho, and Samuel R. Bowman. 2021. Comparing test sets with item response theory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1141–1158, Online. Association for Computational Linguistics.

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*.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024. Helpsteer 2: Open-source dataset for training top-performing reward models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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

Lucas Weber, Elia Bruni, and Dieuwke Hupkes. 2023a. The icl consistency test. *ArXiv preprint*, abs/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, Singapore. Association for Computational Linguistics.

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. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan 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 *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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 *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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*, abs/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.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. 2024. Advancing llm reasoning generalists with preference trees. *Preprint*, arXiv:2404.02078.

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*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2024. Jasper and stella: distillation of sota embedding models.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *ArXiv preprint*, abs/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 *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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, Torino, Italia. ELRA and ICCL.

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. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan 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 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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.

Yan Zhuang, Qi Liu, Yuting Ning, Weizhe Huang, Rui Lv, Zhenya Huang, Guanhao Zhao, Zheng Zhang, Qingyang Mao, Shijin Wang, and 1 others. 2023. Efficiently measuring the cognitive ability of llms: An adaptive testing perspective.

A Appendix

A.1 Instruction Categorization – details

We present the zero-shot prompt for classifying instructions in Figure 19. As for training the SetFit classifier, we leverage the training data detailed in Table 12, 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 Generating $\mathcal{D}_{\text{diff}}$

Figure 8 shows the proportions of different instruction-response pairs that we use as the training data for the difficulty scorer. In Table 13 we report the benchmark data sources from which we obtained the data, as well as the type of evaluation metric that we use to evaluate the 18 LLMs from Table 4.

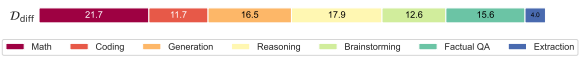


Figure 8: Proportions of different categories in the difficulty scorer training set $\mathcal{D}_{\text{diff}}$ after cleaning.

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) as samples 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., 2023). For the LLM-as-a-judge evaluation, we use GPT4o as the judge and evaluate only a subset of 4 LLMs from Table 4 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).

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 4: 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 5: Hyperparameter details for training the difficulty scorer

Figure 9 demonstrates the preprocessing step of calculating the deviation from the average as mentioned in Section 3.2.1.

A.2.2 Difficulty scorer training

We equip regular CausalLLMs with a regression head by pooling the final hidden states and adding a linear projection to a scalar output, then finetune these models on the difficulty scoring task using the hyperparameters detailed in Table 5. We evaluate four different base models as difficulty scorer—Llama-3.1-8B (Grattafiori et al., 2024), Qwen-2.5-7B (Team, 2024a), Qwen3-4B and Qwen3-8B (Team, 2025))—and select Qwen3-8B, which achieves the best performance on in-distribution evaluation data.

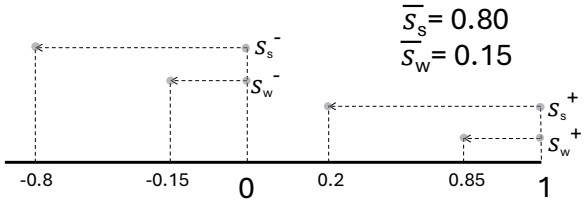


Figure 9: An illustration of calculating the *deviation of the average* per dataset: Assuming we have a strong and a weak model with the average scores of $\bar{s}_s = 0.8$ and $\bar{s}_w = 0.15$ respectively. We subtract these averages from the respective item scores (here two examples with $s^+ = 1$ and $s^- = 0$). We can see how an incorrect response of the strong model produces a much more negative score than an incorrect response from the weak model (compare s_s^- and s_w^-). In other words, when a strong model gets an item wrong, this is much more meaningful for its difficulty compared to the weak model. Similarly, when a weak model correctly solves a datapoint, it is a much more meaningful signal for the data point's ease than when a strong model is correct.

A.2.3 Effectiveness

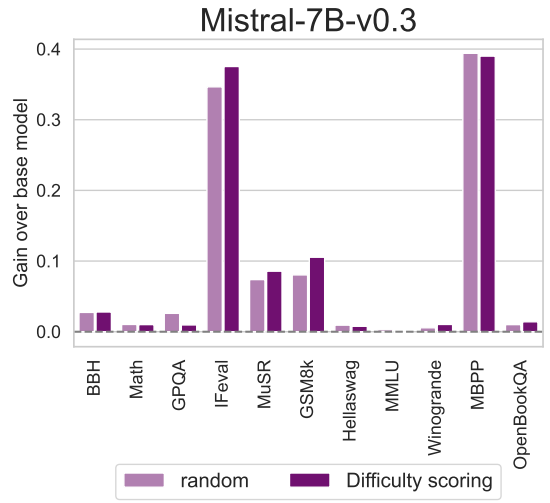
To evaluate the effectiveness of the difficulty scorer for data selection across benchmarks, we sample 25k data points from the datasets in Table 2, either at random or based on LASER *difficulty score* ranking. The results in Figure 10a show that difficulty-based sampling outperforms or is comparable to random sampling on all benchmarks except GPQA. For comparison, Figure 10b reports results using the DEITA *complexity scorer* under the same setup.

A.2.4 Validity

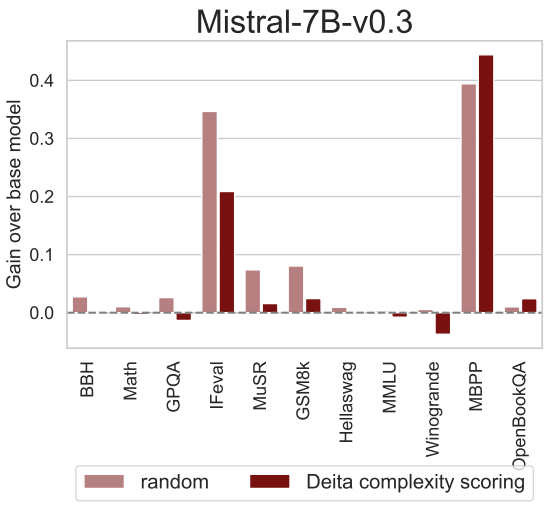
We further evaluate whether the difficulty scores are akin to the notion of difficulty in humans or whether our scorer fits a spurious feature of our training data.

We first test it against instruction length, a common spurious feature of difficulty/complexity scorers. While difficult instructions are plausibly often longer than easier instructions, there is no causal relationship between the length and difficulty. 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 11 how our difficulty scorer is weakly related to the spurious feature of instruction length, while the correlation of DEITA complexity is much higher.

Besides showing how the difficulty scorer does not rely on instruction length, we also use an external criterion to evaluate whether it measures difficulty in the human sense. For this end, we score the



(a)



(b)

Figure 10: (a) Performance comparison of difficulty scorer vs. random sampling; (b) Performance comparison of complexity scorer vs. random sampling

CodeForces section of DeepMind's code-contests dataset (Li et al., 2022) with our difficulty scorer as well as the DEITA complexity scorer and correlate the scores with the given cf_rating. Figure 12 shows how the difficulty scorer is much better at predicting human difficulty scores than the DEITA complexity scorer.

A.2.5 Experimentation with IRT-based targets

We further experimented with difficulty targets grounded in psychometric methods called item re-

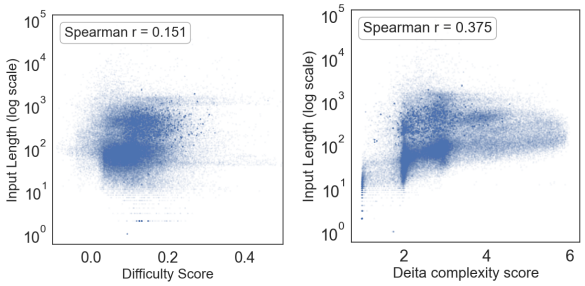


Figure 11: Relation between complexity/difficulty scores and length of the input sequence. The length of the input sequence should, in most cases, be unrelated to its difficulty.

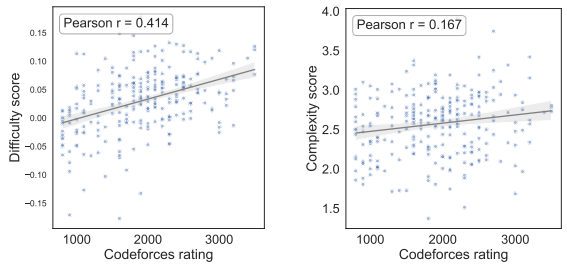


Figure 12: Relation between complexity/difficulty scores and the human code-forces difficulty scores. Difficulty scores show higher correlation to human scores than complexity scores.

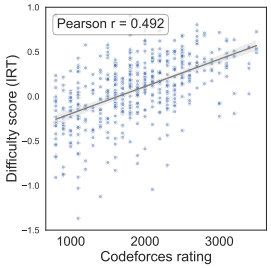


Figure 13: Correlation of difficulty scores with Codeforce ratings when the difficulty scorer is trained on IRT βs as targets (as described in Appendix A.2.1). As we can see here, IRT βs yields higher validity with the external criterion. However, we find it not to benefit data selection.

sponse theory (IRT; see Lord, 1968; Brzezińska, 2020; Van der Linden, 2018; Cai et al., 2016). IRT provides a framework for estimating latent traits from observed responses and has recently been applied to the evaluation of large language models (see e.g. Lalor et al., 2016; Rodriguez et al., 2021; Vania et al., 2021; Zhuang et al., 2023; Polo et al., 2024a). In our setting, an IRT model estimates item-level parameters from correctness scores across the LLMs from Table 4. We fit such

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 6: Performance of 10 considered LLMs on IFEval.

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 7: 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.

a model separately on each dataset in $\mathcal{D}_{\text{diff}}$ and use the resulting item difficulty parameters β as targets for training our difficulty scorer. We find that the IRT-based difficulty scorer yields improved results on the correlations with the Codeforces ratings (the validity criterion; see Figure 13). At the same time, this IRT-based difficulty scorer does not yield better results when used for data selection. We see the use of IRT-based difficulty scorers as an exciting future research direction, as the possibilities to gain more detailed insights into data point features are great (e.g. by using 2PL or 3PL instead of 1PL models or by extending from unidimensional to multidimensional models).

A.3 Quality Scorers – details

A.3.1 Instruction-following Scorer

Prompts used by LLM-annotator/judge for deriving instruction-following scores can be found in Figure 21, 22 and 23. As our test dataset, we collected responses from 10 LLMs on the IFEval benchmark (see Table 6), available on open-llm-leaderboard's dataset collection of evaluation details.[9] Table 7 presents the evaluation results of various LLMs as the annotator/judge.

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

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 8: 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.

A.3.2 Code-quality Scorer

The prompt used by LLM-annotator/judge for deriving code-quality scores can be found in Figure 20. Table 8 presents the evaluation results of various LLMs as the code reviewer.

A.3.3 Effectiveness

We test the efficacy of LASER *task-specific quality scorers* against the general-purpose DEITA *quality scorer*, by comparing the performance of Mistral-7B-v0.3 finetuned on 25k data points, either randomly selected from data in Table 2, or based on ranking by considered quality scorers. For each considered benchmark, we selected only samples from the relevant category as defined in Table 3. Figure 14 demonstrates the clear gain of process reward model (PRM) scoring for *Math* benchmarks, particularly GSM8K. For *Coding* benchmarks, code-quality scoring outperforms DEITA quality scoring on HumanEval, but achieves similar results as random. IF-quality scoring surpasses both DEITA quality scoring and random on *Generation/Brainstorming* benchmarks. For other categories (*Reasoning*, *Factual QA* and *Extraction*), DEITA quality scoring provides noticeable benefits only on TruthfulQA, with minimal impact on other benchmarks.

A.4 Sampling – details

A.4.1 Effectiveness

In Section 3.3, we introduce a sampling technique that aims to improve the diversity of the target dataset $\mathcal{D}'$. The sampling technique consists of two components: firstly, we stratify the sampling by the categories that we assigned using the instruction classifier π_c (according to the proportions shown in Figure 2 [**laser**]) and, secondly, we cluster the datapoints in embedding space and restrict the sampling to one datapoint per cluster. We sample here once 25k data points with the first technique (which we will call *random+*) and once 25k data points with both methods (correspondingly, *random++*) from $\mathcal{D}$. The *random* in the method name indicates that we select samples otherwise randomly (hence, not using any scoring during sampling). We then train a Mistral-7B-v0.3-base model on the resulting datasets and compare the results with entirely random sampling in Figure 16.

A.4.2 Efficiency

In Table 9, we compare the sampling runtime of LASER with DEITA, which iteratively builds $\mathcal{D}'$ by adding dissimilar samples based on embedding distance, as well as CAR, which runs clustering on the entire dataset $\mathcal{D}$. While DEITA is highly efficient given small m (e.g., 1k), its runtime grows rapidly with larger m. Meanwhile, CAR suffers from out-of-memory issues at large m. In contrast, LASER remains stable across different m and is generally more efficient than CAR.

m	DEITA	CAR	LASER
1k	92.3 s	5515.7 s	1505.1 s
10k	970.2 s	5750.9 s	1555.9 s
100k	41055.3 s	OOM	2818.1 s

Table 9: Runtime comparison for different m.

A.5 Datasets – details

A.5.1 Datasets main experiments

Figure 18 shows SetFit classification results for IT datasets from which we composed $\mathcal{D}$. On the other hand, Figure 15 shows which of the source datasets are sampled when we use various sampling strategies. Interestingly, while LASER maintains relative diversity across different source datasets, DEITA is completely dominated by only two datasets: microsoft/orca-agentinstruct-1M-v1 (200k sampled) and WizardLMTeam/WizardLM_evol_instruct_V2_196K.

A.5.2 Datasets robustness experiments

Dataset details $\mathcal{D}_{\text{strong}}$ and $\mathcal{D}_{\text{weak}}$ In Section 4.2.2, we introduce $\mathcal{D}_{\text{strong}}$ and $\mathcal{D}_{\text{weak}}$. While $\mathcal{D}_{\text{strong}}$ simply consists of the Tülu v3 dataset, $\mathcal{D}_{\text{weak}}$ is a aggregation of different datasets. The single components exhibited either in our own pilot experiments or in prior related work comparably weak performance to other datasets. This weaker performance also shows, when we compare the finetuning using full $\mathcal{D}_{\text{weak}}$ with the finetuning us-

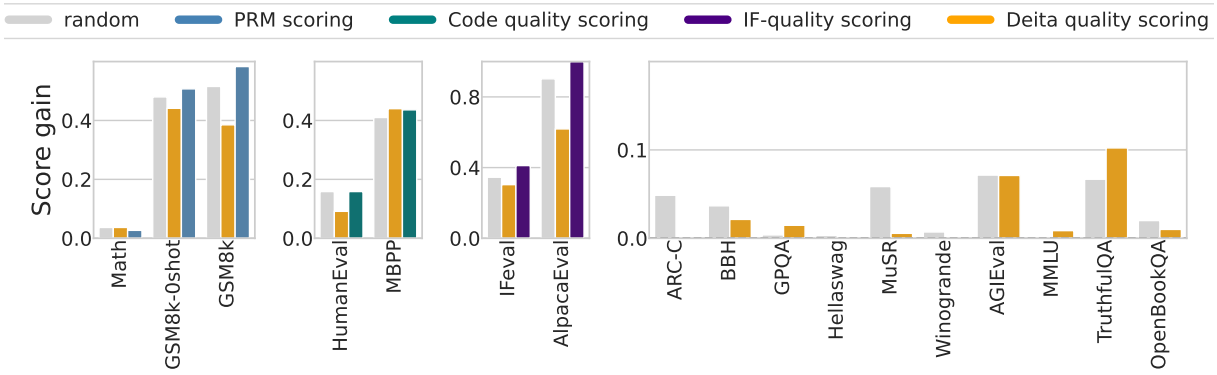


Figure 14: Performance of quality scoring-based sampling on various benchmarks.

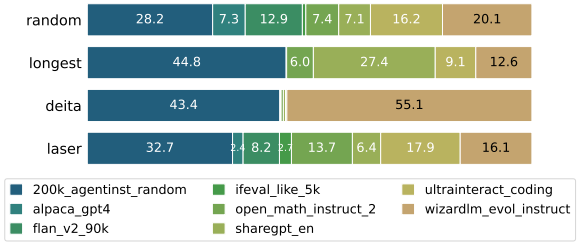


Figure 15: Source dataset proportions of different sampling strategies for 100k samples.

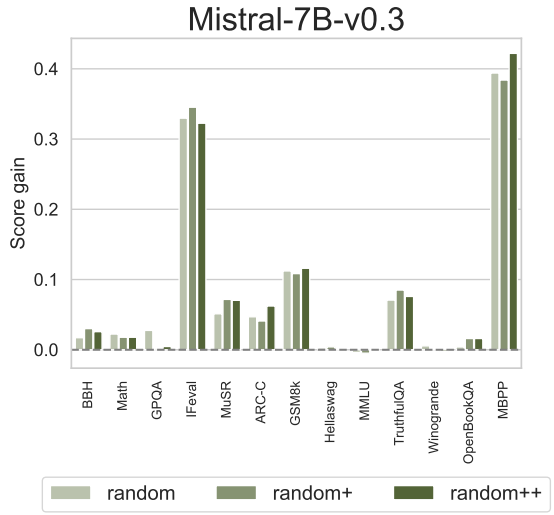


Figure 16: Effect of different sampling strategies on training outcomes. While *random* selects $m = 25$k arbitrary samples, *random+* samples by stratifying the categories in D' according to the proportions shown in Figure 2 (**laser**). Ultimately, *random++* also includes clustering (as described in Section 3.3, while randomly selecting a datapoint from each cluster)

ing full $\mathcal{D}_{\text{strong}}$ in Figure 6, with the latter significantly outperforming the former.

Dataset	**#samples** (*#turns*)
ConiferLM/Conifer (Sun et al., 2024)	13k
databricks/databricks-dolly-15k (Conover et al., 2023)	15k
akoksal/LongForm (Köksal et al., 2023)	23k
alpaca (Taori et al., 2023)	52K
vicgalle/alpaca-gpt4	52K
ai2-adapt-dev/flan_v2_converted (Sanh et al., 2022)	90K
nvidia/Daring-Anteater (Wang et al., 2024)	100k
AI-MO/NuminaMath-CoT[10] (Li et al., 2024a)	250k
all	595K

Table 10: Composition of $\mathcal{D}_{\text{weak}}$

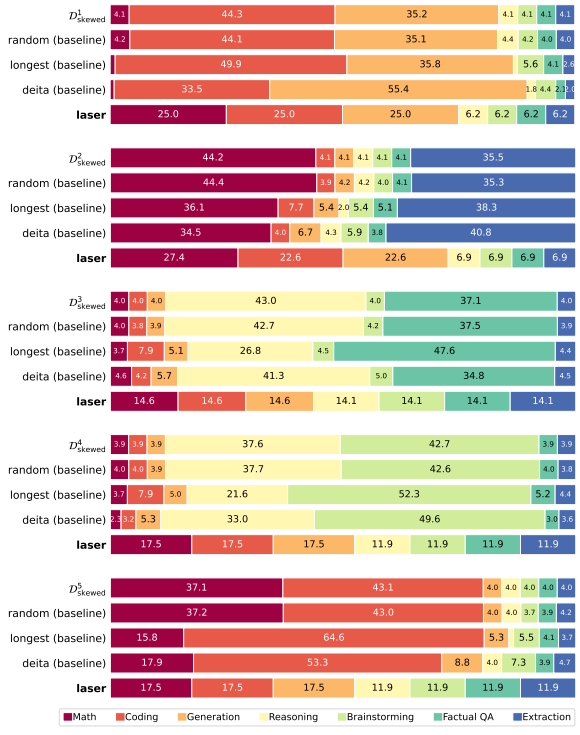


Figure 17: Category distribution for datasets sampled from the 5 bias source datasets $\mathcal{D}_{\text{skewed}}$ using different sampling strategies.

Results of experiments with $\mathcal{D}_{\text{skewed}}$ In Section 4.2.2, we experiment with skewed data distributions. In Figure 17, we show the resulting category distributions when sampling with different sampling approaches from the 5 skewed source distributions. As we can see, only LASER reduces the introduced bias in the skewed source data, while DEITA is not effective in reducing the bias, but in many cases exacerbates it.

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 11 details the hyperparameters used for finetuning.

A.7 Results – details

Figure 24 shows the scaling experiment results across benchmarks, reported without normalizing *score gain*.

Hyperparameter	falcon-10B	llama-8B	mistral-7B	qwen-3B	smollm-1.7B
batch_size	4	8	8	8	32
gradient_accumulation	16	8	16	8	16
learning_rate	2.0e-05	5.0e-06	5.0e-06	2.0e-05	1.0e-04
num_train_epochs	2	2	2	2	3
weight_decay	0.1	0.01	0.01	0.01	0.01
warmup_ratio	0.03	0.03	0.1	0.03	0.03
lr_scheduler_type			cosine		
attn_implementation			flash_attention_2		
max_seq_length			2048		
neftune_noise_alpha			5		
use_liger			True		

Table 11: Hyperparameter details for finetuning.

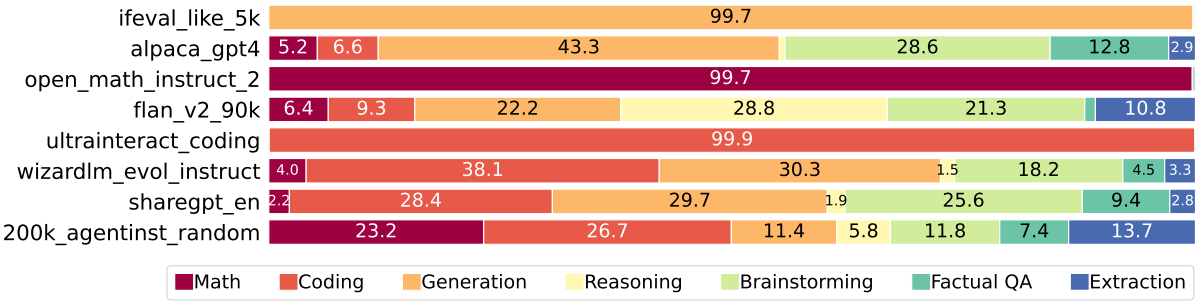


Figure 18: Category distribution in IT datasets used in the experiments.

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 12: 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., 2020)	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 13: Datasets used in difficulty scorer training.

```
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 19: 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 ëview", "final_verdict", "code_original" and "code_revision" as keys.

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

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

Figure 20: 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 21: 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 22: 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 23: Zero-shot prompt for response evaluation.

(a) AGIEval (Zhong et al., 2024) (b) AlpacaEval (Dubois et al., 2024) (c) ARC-Challenge (Clark et al., 2018)

(d) BBH (Suzgun et al., 2023) (e) GPQA (Rein et al., 2024) (f) HellaSwag (Zellers et al., 2019)

(g) GSM8k$_{\text{cot}}$ (Cobbe et al., 2021) (h) GSM8k$_{\text{cot-0-shot}}$ (Cobbe et al., 2021) (i) Math (Hendrycks et al., 2021c)

(j) HumanEval (Chen et al., 2021) (k) MBPP (Austin et al., 2021) (l) IFeval (Zhou et al., 2023b)

(m) MMLU (Hendrycks et al., 2021b) (n) MuSR (Sprague et al., 2024) (o) OpenBookQA (Mihaylov et al., 2018)

(p) TruthfulQA (Lin et al., 2022) (q) Winogrande (Sakaguchi et al., 2020)

Figure 24: Results scaling across all benchmark datasets