When Data is the Algorithm: A Systematic Study and Curation of Preference Optimization Datasets

Aladin Djuhera
Technical University Munich
aladin.djuhera@tum.de

Farhan Ahmed
IBM Research
farhan.ahmed@ibm.com

Swanand Ravindra Kadhe
IBM Research
swanand.kadhe@ibm.com

Syed Zawad
IBM Research
szawad@ibm.com

Heiko Ludwig
IBM Research
hludwig@ibm.com

Holger Boche
Technical University Munich
boche@tum.de

Abstract

Aligning large language models (LLMs) is a central objective of post-training, often achieved through reward modeling and reinforcement learning methods. Among these, direct preference optimization (DPO) has emerged as a widely adopted technique that fine-tunes LLMs on preferred completions over less favorable ones. While most frontier LLMs do not disclose their curated preference pairs, the broader LLM community has released several open-source DPO datasets, including TuluDPO, UltraFeedback, ORPO, HelpSteer, and Code-Preference-Pairs. However, the construction process behind these datasets often lacks valuable metadata, design rationales, and quality annotations. This missing context makes it difficult to understand how preferences were selected, what task types they span, and how well they reflect human judgement on a per-sample-level. In this work, we present the first comprehensive, data-centric analysis of open-source DPO corpora. We leverage the Magpie framework to annotate each sample for task category, input quality, and preference reward, a reward-model-based signal that validates the preference order without relying on human annotations. This enables a scalable, fine-grained inspection of preference quality across datasets, revealing structural and qualitative discrepancies in reward margins. Building on these insights, we systematically curate a new DPO mixture, **UltraMix**, that draws selectively from all five corpora while removing noisy or redundant samples. UltraMix is 30% smaller than the best-performing individual dataset yet exceeds its performance across key benchmarks. We publicly release all annotations, metadata, and our curated mixture to facilitate future research in data-centric preference optimization.

1 Introduction

Aligning large language models (LLMs) after pre-training is an important step in ensuring that models behave helpfully and according to pre-defined preferences [1, 2]. This post-training phase refines the base model's behavior through additional data and supervision, guiding it toward more desirable outputs to ensure instruction following, step-by-step reasoning, or adhering to task-specific constraints [3, 4]. Alignment can be approached through several complementary lenses[1]: supervised fine-tuning (SFT), reward modeling, and reinforcement learning (RL) [5]. Among RL-based approaches, two methods have gained particular prominence: Proximal Policy Optimization (PPO) [6] and Direct Preference Optimization (DPO) [7]. DPO, in particular, has emerged as a favored technique due to its

[1] See App. A for more details.

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

efficiency and simplicity: models are directly trained to prefer one completion over another using curated preference pairs without requiring a reward model or policy rollouts.

However, proprietary LLMs often leverage vast internal corpora for both SFT and DPO, datasets that remain inaccessible to practitioners and researchers, often because they are not disclosed or have licensing restrictions. Nevertheless, the open-source community has made notable progress by releasing publicly available DPO datasets, including *TuluDPO* [8], *UltraFeedback* [9], *ORPO* [10], *HelpSteer* [11], and *Code-Preference-Pairs* [12]. These corpora have become essential building blocks for many instruction-tuned open models and contribute meaningfully to the broader reproducibility efforts in LLM preference tuning.

While the coarse composition behind these datasets is often well-documented, they remain opaque on a sample-level: metadata is missing, design rationales are loosely documented, and quality annotations are largely absent. This missing context poses several challenges: First, preference mixtures frequently depend on other datasets (often recursively or implicitly), making it hard to trace provenance, tag overlaps, or assess originality. For instance, TuluDPO inherits samples from UltraFeedback, but without fine-grained labels on task categories, it is difficult to isolate or reuse subsets tailored for math, instruction following, or reasoning tasks. Second, only a few datasets such as HelpSteer provide explicit human annotated preference scores to justify chosen completions, while others provide only binary pairs without ranking information, making it unclear how much better the preferred sample actually is. Finally, DPO datasets often specialize in narrow domains (e.g., code generation or math), and comprehensive mixtures spanning multiple instruction types remain rare.

Inspired by recent work from Djuhera et al. [13], which analyzed and curated SFT datasets using Magpie [14] annotations, we turn our attention to DPO datasets with the goal of bringing diagnostic rigor, transparency, and reproducibility to preference optimization corpora. In particular, we use Magpie to generate *preference rewards*, an independent, reward-model-based signal that helps evaluate whether chosen completions are indeed better justified than discarded ones. This allows us to assess preference signals at scale and to validate the reliability of the preference order, particularly when human annotations are unavailable or inconsistent. Our main contributions are as follows:

- **Large-Scale Evaluation**: We present the first systematic cross-analysis of five open DPO datasets: TuluDPO, UltraFeedback, ORPO, HelpSteer, and Code-Preference-Pairs, spanning both general-purpose and domain-specific tasks. We conduct preference fine-tuning on two SFT models, Tulu3-8B-SFT [8] and SmolLM2-1.7B-SFT [15], and evaluate performance across 14 benchmarks from popular Open LLM Leaderboards [16, 17]. By holding all training parameters constant, we allow for a fair comparison between datasets.
- **Sample-Level Analysis**: Using Magpie [14], we annotate each preference pair for task category, difficulty, input quality, and preference reward, allowing us to both validate and quantify chosen and discarded completions. In our extensive analysis, we reveal substantial variation in dataset composition, quality, and alignment performance, both across tasks and model scales.
- **New DPO Mixture**: Based on our analysis, we construct **UltraMix**, a high-quality DPO mixture that draws from all five datasets and is systematically curated on a per-sample-level to retain only high-quality instructions, high-utility task categories, and coherent preference rewards, removing redundant or noisy samples in the process. UltraMix is 30% smaller than TuluDPO and consistently achieves better performance on key benchmarks while improving task diversity.

To support ongoing research in preference optimization, we will publicly release our code and annotations for TuluDPO, UltraFeedback, ORPO, HelpSteer, Code-Preference-Pairs, and UltraMix.

2 Background and Motivation

In DPO, the dataset and the quality of its preference pairs are critical not only for instilling consistent behavior but also for enhancing downstream performance. To this end, practitioners often either combine entire DPO datasets or select subsets from multiple sources in addition to separately curated preference pairs. However, a principled approach to constructing effective DPO mixtures would require a comprehensive, side-by-side comparison of existing datasets, their sample qualities, and task categories, a need that remains largely unmet. This is partly due to the computational overhead associated with DPO training, especially at scale, as well as the lack of fine-grained annotations.

In this work, we present the first such analysis across five widely used DPO datasets for two representative models. We evaluate performance across scales and identify dataset-specific trade-offs.

TuluDPO. Lambert et al. [8] curated TuluDPO for post-training Llama-3.1 models [18]. The dataset consists of 272,898 preference pairs drawn from several sources, consolidating on- and off-policy preference data from over seven constituent datasets, including UltraFeedback and WildChat [19]. While TuluDPO represents one of the most comprehensive, general-purpose DPO corpora available, its construction inherits samples from several upstream mixtures without per-sample metadata, making fine-grained analysis and targeted reuse difficult.

UltraFeedback. Cui et al. [9] released curated preference pairs for over 60,000 prompts, spanning instruction following, truthfulness, honesty, and helpfulness. Instructions were sourced from high-quality public datasets such as FLAN [20] and their completions were sampled from 17 diverse LLMs, including GPT-4 [21]. While UltraFeedback does provide scalar preference scores (assessed via GPT-4), they remain inconsistent, for example, assigning the same reward score for both chosen and discarded responses. This can occur when multiple responses demonstrate similar performance and the preference pair is thus selected arbitrarily.

ORPO. Labonne [10] curated ORPO as a mixture of preference pairs designed for training with Optimized Rejection-based Preference Optimization (ORPO) [22] and DPO. It aggregates 44,218 preference pairs from several high-quality sources, particularly from Argilla, with chosen responses filtered to meet minimum reward thresholds. ORPO offers a broad distribution of general, factual, and safety-aligned instructions for training general-purpose preference models.

HelpSteer. Wang et al. [11] curated a high-quality instruction following dataset with human annotations for helpfulness, correctness, coherence, complexity, and verbosity. The small-scale dataset comprises 10,681 prompts where each response is scored by over 1,000 human annotators on a fine-grained Likert-scale, offering multi-dimensional preference signals rather than binary choices. These annotations can be used to construct preference pairs, for example, based on mean scores.

Code-Preference-Pairs. Vezora [12] introduced this synthetic code dataset to train models for bug identification and correction. Each sample contains two code variants: an accepted version with correct functionality and inline explanations, and a discarded version where bugs are surgically introduced without explicit indicators. The dataset spans over 55,000 preference pairs across multiple languages and coding problems, with samples derived from several sources, including Evol-Instruct.

We selected these datasets due to their prominence in open-source community blog posts and dataset hubs [23]. In the following sections, we benchmark the performance of each DPO dataset and conduct an extensive side-by-side comparison of their compositions on a per-sample basis.

3 Benchmarking DPO Datasets

We use *Open-Instruct* [24] to perform DPO training on two representative models: Tulu3-8B-SFT [8] and SmolLM2-1.7B-SFT [15]. These models were SFT-tuned from Llama-3.1-8B [18] and SmolLM2-1.7B, but have not been aligned for preference yet. To assess each DPO dataset's utility, we follow the evaluation setup in Djuhera et al. [13] and use *LM Evaluation Harness* [25] to benchmark downstream performance on 12 representative tasks from Open LLM Leaderboards V1 [16] and V2 [17], as well as on two code generation tasks: HumanEval and HumanEval+ [26]. We use this setup for all of our experiments. Additional details on training and evaluation are provided in App. B.

To provide a unified comparison, we report mean scores over the Leaderboard V1 and V2 benchmarks, as well as an overall average across all 14 benchmarks in Table 1. For both Tulu3 and SmolLM2, TuluDPO outperforms every other DPO dataset on average and on both leaderboard benchmarks, demonstrating it is well curated across a variety of different tasks. ORPO outperforms UltraFeedback on code, math, and reasoning tasks for Tulu3. However, for SmolLM2, UltraFeedback surpasses ORPO on math, reasoning, and instruction following tasks. For both models, HelpSteer falls behind UltraFeedback and ORPO, while surpassing Code-Preference-Pairs, which only shows noticeable gains on code benchmarks. The results further reveal that SmolLM2 is significantly less capable than Tulu3, which can be attributed to its smaller model size and its pre-training and SFT focus on conversational chatbot behavior rather than math, code, and reasoning tasks.

Table 1: DPO results for Tulu3-8B-SFT and SmolLM2-1.7B-SFT trained on TuluDPO, Ultrafeedback, ORPO, HelpSteer, and Code-Preference-Pairs, evaluated on Open LLM Leaderboards (averaged) and code benchmarks. The overall average is across all benchmarks. Best scores (row-wise) are in **bold**.

	Tulu3-8B-SFT						SmolLM2-1.7B-SFT					
Benchmark	SFT	TuluDPO	UltraFB	ORPO	HelpSteer	CodePref	SFT	TuluDPO	UltraFB	ORPO	HelpSteer	CodePref
Knowledge												
MMLU (5-shot)	62.30	**63.47**	62.53	62.31	62.04	59.96	49.46	49.49	**49.63**	48.65	48.50	47.47
MMLU-Pro (5-shot)	28.08	**28.98**	28.41	27.90	27.43	27.53	18.54	**19.76**	19.44	18.98	18.81	18.27
TruthfulQA (0-shot)	46.84	**56.78**	50.26	52.26	47.43	56.15	40.94	**45.73**	45.50	39.18	41.19	44.43
GPQA (0-shot)	28.44	29.61	28.69	29.70	29.36	**29.95**	27.60	28.27	27.94	28.10	**28.61**	25.59
Reasoning												
ARC-C (25-shot)	54.95	**57.93**	56.91	56.91	54.52	45.05	48.89	**50.94**	49.32	48.98	46.84	46.84
BBH (3-shot)	38.59	40.46	39.91	**41.80**	38.15	36.07	35.62	**36.08**	35.48	35.29	35.71	34.65
MuSR (0-shot)	43.25	41.93	41.93	**43.78**	42.06	41.53	33.60	34.09	**34.31**	31.35	32.67	30.48
Commonsense												
HellaSwag (10-shot)	60.41	**64.85**	62.29	60.08	61.20	53.38	52.91	53.42	53.69	**53.98**	52.73	50.23
WinoGrande (5-shot)	**76.40**	75.30	75.53	74.27	76.16	69.93	65.32	**67.01**	66.85	66.93	66.77	62.35
Instruction Following												
IF-Eval (0-shot)	72.45	**80.35**	71.59	71.65	73.93	69.90	50.54	**51.97**	51.76	50.02	45.46	42.79
Math												
GSM8K (5-shot)	76.19	79.48	78.47	**81.96**	76.88	76.35	46.84	48.84	**48.98**	47.99	48.07	44.98
MATH (4-shot)	12.08	**22.66**	18.81	20.39	14.88	15.11	4.46	5.63	**5.87**	5.44	4.08	4.42
Code (pass@1)												
HumanEval	57.93	67.24	61.59	64.63	59.15	**70.68**	1.22	1.83	1.83	1.22	1.22	**2.44**
HumanEval+	43.29	46.36	42.49	41.63	42.93	**50.61**	0.61	1.22	1.22	0.61	0.61	**1.83**
Leaderboards												
Open LLM Leaderboard 1	62.85	**66.30**	64.33	64.63	63.04	60.14	50.73	**52.57**	52.33	50.95	50.68	49.38
Open LLM Leaderboard 2	37.15	**40.66**	38.22	39.20	37.64	36.68	28.39	**29.30**	29.13	28.20	27.56	26.03
Overall	50.09	**53.96**	51.39	52.09	50.44	50.16	34.04	**35.31**	35.13	34.05	33.66	32.63

These findings raise several initial research questions: First, given the general-purpose data included in TuluDPO, can the strengths and weaknesses observed in DPO training be attributed to the distribution of specific task categories and the quality of their associated samples? Second, how can specialized preference datasets such as Code-Preference-Pairs be effectively combined with general-purpose datasets to create higher-performing mixtures on individual benchmarks? Finally, can LLM-based reward models be used to automatically verify the chosen completions in preference pairs, and if so, can we filter out samples where the chosen response is actually inferior to the discarded one? To address these questions, we present a detailed analysis of all of above DPO datasets in the following sections, enabling more informed decisions about dataset composition and mixture strategies.

4 Comparative Analysis of DPO Datasets

We conduct a detailed side-by-side analysis of all considered DPO datasets. Each sample is systematically annotated using the Magpie framework, a customizable self-synthesis annotation pipeline, yielding fine-grained labels for task category, input quality, difficulty, response quality, language, and safety. These structured annotations enable evaluation of preference pairs on a per-sample-level, allowing us to judge the preference order and correlate samples with quality and difficulty indicators. This detailed characterization forms a principled foundation for constructing effective DPO mixtures.

4.1 Annotating DPO Samples using Magpie

Magpie leverages an LLM as a judge to assign labels for *task category* (12 classes), *input quality* (rated from very poor to excellent), *query difficulty* (rated from very easy to very hard), *safety* (assessed via Llama-Guard 2 [27]) and *language*, using specialized prompt templates (see App. C). To assess *preference rewards*, we further evaluate the *response quality* of chosen and discarded completions using the Llama-3-8B-Instruct-based reward model introduced by Dong et al. [28], which is fine-tuned using the Bradley-Terry formulation on a diverse mixture of high-quality preference datasets, including HH-RLHF [29] and SHP [30]. This particular reward model has proven to achieve strong alignment with human preference judgments, making it a reliable signal for quantifying how well the model completions satisfy the instruction. For all other labels, we use Llama-3.3-70B-Instruct [18] as the judge model given its demonstrated reliability, particularly for quality and difficulty annotations, as assessed by Djuhera et al. [13]. In the following analysis, we focus on task category, query difficulty, input quality, and preference rewards as the primary signals for dataset comparison, and refer to App. D.5 for additional details on low-impact language and safety distributions.

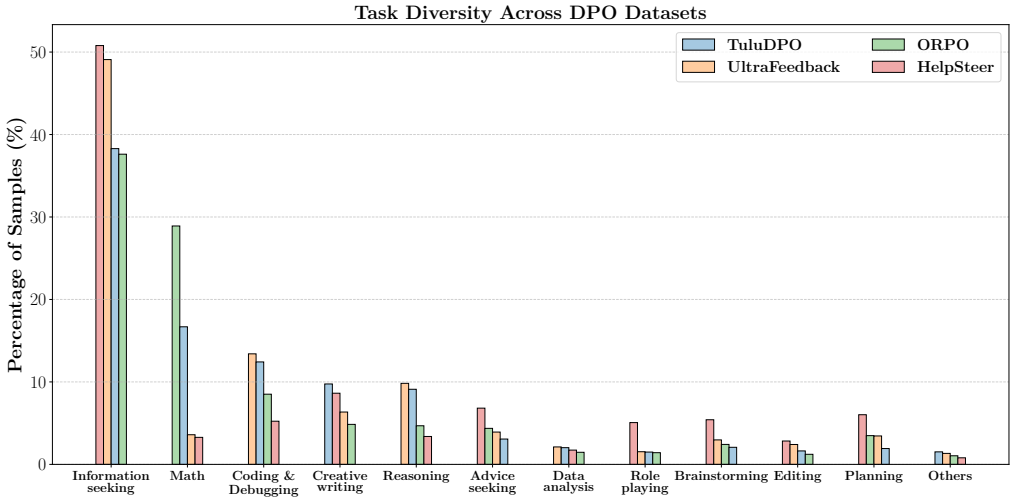
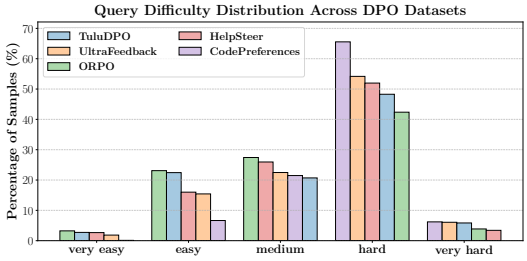
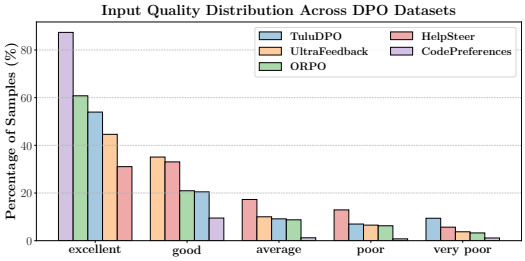


Figure 1: Task diversity across all datasets as annotated by Magpie (excluding Code-Preference-Pairs). Bars indicate the proportion of each dataset dedicated to different task categories. Information seeking dominates across all datasets, followed by math and coding. See App. D.1 for more details.



(a) Query difficulty: most instructions are labeled as hard or medium, suggesting that both general-purpose and domain-specific DPO datasets tend to include challenging prompts to encourage stronger alignment.



(b) Input quality: TuluDPO, UltraFeedback, ORPO, and Code are mostly labeled as good or excellent, indicating well-formulated prompts. HelpSteer contains a non-negligible portion of low-quality inputs.

Figure 2: DPO prompt analysis: (a) Distribution of query difficulty. (b) Distribution of input quality.

4.2 Task Categories, Query Difficulty, and Input Quality

The distribution of task categories in Fig. 1 shows that information seeking tasks dominate across all DPO datasets, mostly involving prompts that ask for detailed explanations, context-specific summaries, or factual clarifications that are typical for instruction following behavior. This trend aligns with the objectives of general-purpose preference fine-tuning and is especially prevalent in HelpSteer and UltraFeedback, where information seeking prompts account for 51% and 49% of the data, respectively. TuluDPO and ORPO also exhibit strong coverage in this category, with 38% and 37% of samples. Math is the second most prominent category with ORPO having the highest proportion at 29%, followed by TuluDPO at 17%. Interestingly, despite its smaller relative proportion, TuluDPO achieves stronger performance on math benchmarks (see Table 1), suggesting that absolute sample count and potentially higher quality samples may play a more critical role in math tuning. Coding, creative writing, and reasoning are similarly represented among UltraFeedback and TuluDPO, while more conversational task types such as editing, role playing, brainstorming, and planning are typically underrepresented across all DPO datasets, each comprising only 2–6% of the data. HelpSteer shows a relatively even distribution outside information seeking, though its limited size restricts its overall impact on downstream performance, falling behind others. We exclude Code-Preference-Pairs from this analysis, as it consists solely of coding samples by design. Overall, these findings indicate that DPO datasets place strong emphasis on instruction following through factual information seeking and structured reasoning, most notably in the domains of math and code.

We show the corresponding distribution of query difficulty in Fig. 2a, revealing that most prompts are labeled as *"hard"* or *"medium"*, with a smaller fraction categorized as *"easy"*. This suggests that preference tuning across both general-purpose and domain-specific datasets tends to favor more challenging instruction-response pairs, which may help induce stronger alignment and improved downstream performance. Fig. 2b further shows the distribution of input quality for all datasets, showing that TuluDPO, UltraFeedback, ORPO, and Code-Preference-Pairs contain predominantly high-quality prompts, with more than 75% rated as either *"good"* or *"excellent"*. This reflects the filtering processes employed during curation. In contrast, HelpSteer, being more broadly distributed in topic and style, contains a non-negligible portion of 35% of prompts rated as *"average"*, *"poor"*, or *"very poor"*, indicating either lack of context or underspecified user intent. We provide additional insights into how query difficulty and input quality vary by task category in App. D.2 and App. D.3.

Our Magpie annotations in Fig. 1 and Fig. 2 reveal several shared patterns across the examined datasets: (a) preference alignment is most effectively instilled through factual information seeking, math, and code, (b) stronger alignment is elicited by more challenging instructions, and (c) quality matters as most high-performing DPO datasets contain precise prompts to maximize alignment signal.

4.3 Preference Reward Analysis

As we compute preference rewards using an independent reward model, it also serves as a verification mechanism for existing reward annotations found in UltraFeedback and HelpSteer. To this end, we use the specialized reward model to first evaluate whether the chosen completion in each preference pair indeed receives a higher reward than the rejected one. Surprisingly, we observe a non-negligible number of cases where the reward model assigns a higher score to the discarded response. Fig. 3 shows, for each DPO dataset, the proportion of samples where the chosen completion is correctly preferred according to the reward model. TuluDPO, UltraFeedback, ORPO, and HelpSteer exhibit similar patterns, with only 70–80% of samples aligning with the chosen preference. This supports our earlier statement that preference decisions may sometimes be arbitrary or based on near-identical completions. In addition, this shows that GPT-4-based reward annotations in UltraFeedback, as well as the simplistic use of mean scores in HelpSteer, are often misaligned with the judgments of a dedicated reward model such as the one used in our analysis. Interestingly, Code-Preference-Pairs appears less affected by such inconsistencies. This may be due to more salient signals (e.g., missing comments or syntax errors) in code tasks, making it easier for both humans and reward models to distinguish between preference pairs.

To further evaluate the reliability of our reward model, we compute the difference between preference rewards of chosen and rejected completions for each sample, and present the resulting distribution in Fig. 4. The histogram reveals that for more comprehensive datasets such as TuluDPO, UltraFeedback, and ORPO, the reward differences are more broadly distributed, reaching the positive tails and indicating a clearer separation between good and bad completions. Code-

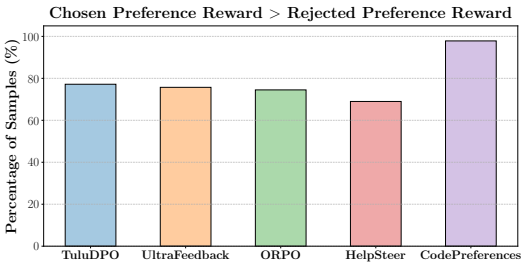


Figure 3: Proportion of samples where chosen completions are correctly preferred.

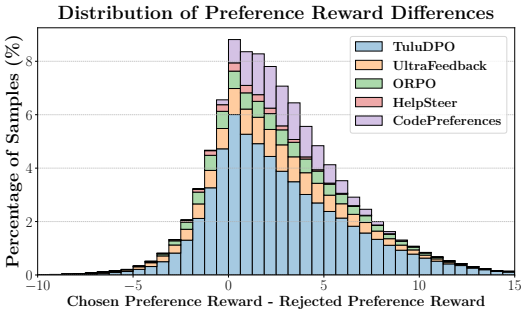


Figure 4: Histogram of reward differences between chosen and rejected completions.

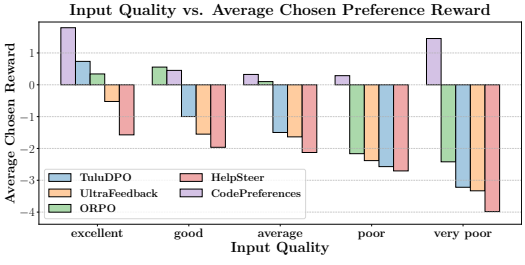


Figure 5: Average reward assigned to chosen completions across different input quality levels.

Preference-Pairs similarly extends to the positive tail but also exhibits smaller differences, particularly for examples where the code is functionally identical but the chosen completion includes additional commentary, making the distinction more subtle. In contrast, HelpSteer shows a distribution centered more closely around zero, suggesting that most of its preference pairs consist of similarly rated completions. This further supports our earlier observation that HelpSteer may provide weaker alignment signals due to less high-quality input signals.

To further examine the relationship between input quality and preference rewards, we analyze the average reward assigned to chosen completions across different input quality levels in Fig. 5. The results reveal a clear upward trend for most datasets: higher input quality is associated with higher preference rewards. This suggests that model completions are strongly correlated with the quality of the prompt, providing, to our knowledge, the first empirical evidence that poorly written instructions may lead to subpar preference alignment. Interestingly, Code-Preference-Pairs exhibits a notable outlier for inputs rated as *"very poor"*. Upon further inspection, many of these prompts contain either contradictory requirements or missing information, but completions still produce fully functional responses by falling back on reasonable assumptions or using dummy examples.

From this analysis, we conclude that our preference reward annotations uncover several non-trivial insights into dataset quality and curation practices. Most notably, our independent reward model is able to identify potentially misaligned preference pairs. Further, our findings highlight that both input quality as well as the difference in preference rewards between chosen and discarded completions emerge as useful filtering criteria, motivating the construction of cleaner and potentially more performant DPO mixtures by selectively retaining samples with high-quality prompts and well-aligned preferences. We refer to App. D for additional results and insights.

5 Using Annotations to Design Reward-Based Data Curation Recipes

We leverage our annotations of input quality and preference rewards to curate a new DPO mixture that draws from all five examined datasets. To this end, we design a set of filters and evaluate their effectiveness through ablation experiments. Our objective is to construct a new mixture that surpasses the current best-performing TuluDPO by selectively retaining only high-quality samples whose preference rewards are aligned with the judgments of our independent reward model.

5.1 Quality- and Reward-Based Curation Recipe

Recipe. We curate an initial mixture by selecting from each dataset only those samples where (a) input quality is high (*"excellent"* or *"good"*), (b) difficulty is greater than *"very easy"* (due to low correlation with downstream performance), and (c) the chosen preference reward is higher than the discarded one (ensuring alignment with our reward model). From each filtered dataset, we then retain samples above the 25th percentile of preference rewards for TuluDPO, UltraFeedback, ORPO, and HelpSteer, and apply a stricter threshold at the 80th percentile for Code-Preference-Pairs to avoid over-representing code-related data. To further reduce redundancy, we also perform deduplication, as TuluDPO contains a substantial overlap with UltraFeedback. This recipe yields a mixture of 170k samples (37% smaller than TuluDPO), which we refer to as *UltraMix-170k*.

Performance Analysis. Table 2 compares evaluation results for UltraMix-170k (UM-170k) against TuluDPO. For Tulu3, UltraMix-170k marginally outperforms TuluDPO on overall scores and Open-LLM leaderboard metrics. This improvement is driven primarily by a substantial increase on TruthfulQA from 56.78% to 61.45%, as well as gains on BBH reasoning. However, UltraMix-170k underperforms on key benchmarks, particularly on both code tasks, MATH, and IFEval, indicating that the curation process could be further refined to boost performance in these areas. For SmolLM, UltraMix-170k tends to underperform TuluDPO in both the overall average and individual leaderboard scores, again with notable gains only on TruthfulQA, where UltraMix-170k improves performance from 45.73% to 48.10%. This suggests that smaller models are more sensitive to data selection.

Given that most LLM benchmarks heavily emphasize coding, math, and instruction following tasks, our initial curation strategy may be too simplistic, not capturing enough samples required for strong downstream performance on code and math benchmarks. In particular, filtering samples based solely on high preference rewards may have removed valuable instruction following examples, especially

Table 2: DPO results for Tulu3-8B-SFT and SmolLM2-1.7B-SFT trained on our curated DPO mixtures UltraMix-170k (UM-170k), UltraMix-187k (UM-187k), and UltraMix-190k (UM-190k), compared to TuluDPO on Open LLM Leaderboards (averaged) and code benchmarks. The overall average is across all benchmarks. Best scores (row-wise) are in **bold**.

	Tulu3-8B-SFT					SmolLM2-1.7B-SFT				
Benchmark	SFT	TuluDPO	UM-170k	UM-187k	UM-190k	SFT	TuluDPO	UM-170k	UM-187k	UM-190k
Knowledge										
MMLU (5-shot)	62.30	63.47	63.27	64.19	**64.61**	49.46	49.49	48.97	49.32	**49.89**
MMLU-Pro (5-shot)	28.08	28.98	28.50	30.24	**30.96**	18.54	19.76	19.31	20.50	**20.87**
TruthfulQA (0-shot)	46.84	56.78	61.45	**63.85**	63.32	40.94	45.73	48.10	**50.38**	50.24
GPQA (0-shot)	28.44	29.61	29.94	31.46	**31.87**	27.60	28.27	27.92	29.85	**30.43**
Reasoning										
ARC-C (25-shot)	54.95	57.93	57.63	58.34	**58.70**	48.89	50.94	48.62	49.82	**50.48**
BBH (3-shot)	38.59	40.46	44.68	44.21	**44.96**	35.62	36.08	35.24	36.24	**36.96**
MuSR (0-shot)	43.25	41.93	42.43	43.42	**44.02**	33.60	34.09	33.22	36.66	**37.14**
Commonsense										
HellaSwag (10-shot)	60.41	**64.85**	63.43	64.02	64.82	52.91	53.42	52.84	54.93	**55.09**
WinoGrande (5-shot)	76.40	75.30	74.98	76.38	**77.06**	65.32	67.01	65.48	67.27	**67.86**
Instruction Following										
IF-Eval (0-shot)	72.45	80.35	79.38	80.19	**81.13**	50.54	51.97	50.80	51.81	**52.54**
Math										
GSM8K (5-shot)	76.19	79.48	79.98	80.93	**82.48**	46.84	48.84	47.96	49.45	**50.14**
MATH (4-shot)	12.08	22.66	21.22	22.03	**23.56**	4.46	5.63	5.35	5.82	**6.05**
Code (pass@1)										
HumanEval	57.93	67.24	65.61	68.06	**69.05**	1.22	1.83	1.83	2.44	**2.44**
HumanEval+	43.29	46.36	45.76	47.67	**48.08**	0.61	1.22	1.22	1.83	**1.83**
Leaderboards										
Open LLM Leaderboard 1	62.85	66.30	66.79	67.95	**68.50**	50.73	52.57	52.00	53.53	**53.95**
Open LLM Leaderboard 2	37.15	40.66	41.02	41.92	**42.75**	28.39	29.30	28.64	30.15	**30.67**
Overall	50.09	53.96	54.16	55.36	**56.04**	34.04	35.31	34.78	36.17	**36.57**

Table 3: Distribution of task categories associated with instruction following capabilities. Our initial UltraMix-170k dataset underrepresents information seeking and reasoning tasks relative to TuluDPO.

Task Category	TuluDPO	UltraMix-170k	UltraMix-187k	UltraMix-190k
Math	5.3%	24.8%	22.1%	19.6%
Coding & Debugging	12.7%	17.3%	14.5%	12.9%
Information seeking	48.6%	28.5%	33.4%	38.7%
Reasoning	19.0%	5.8%	7.9%	9.2%

from TuluDPO, thereby negatively impacting results on these benchmarks. We explore this hypothesis further through a distributional analysis of the resulting mixture per task category.

Task Analysis. Our initial mixture underperforms on benchmarks that heavily rely on instruction following capabilities. This is a critical skill that has been shown to influence performance across a wide range of tasks, as observed by Lambert et al. [8], Djuhera et al. [13], and Cohere et al. [31]. Since the majority of samples in our mixtures originate from TuluDPO (due to its significantly larger size compared to the other DPO datasets), we examine the distribution of instruction following task categories in Table 3. The analysis reveals that UltraMix-170k underrepresents information seeking and reasoning samples, with relative reductions of 20% and 13%, respectively. In contrast, the proportions of math and coding samples have increased. However, as previously noted, performance on these categories tends to depend on absolute sample count than relative proportion. These findings confirm that our quality-based curation recipe would benefit from additional task-aware filtering to preserve a stronger representation of instruction following tasks that increase overall performance.

5.2 Quality-, Reward- and Task-Based Curation Recipe

Adapted Recipe. We extend our previous reward-based curation strategy using two boosting techniques. First, we increase the absolute sample count, particularly for math and code, to improve overall performance. To this end, we augment UltraMix-170k with 25% more samples from TuluDPO, UltraFeedback, ORPO, HelpSteer, and Code-Preference-Pairs, where the chosen preference reward is higher than the discarded one. This yields 17,000 additional samples, resulting in *UltraMix-187k*.

Next, we specifically boost instruction following. Since the current augmentation already includes most very high-quality samples, we select additional information seeking and reasoning examples with preference rewards above the 70th percentile but with slightly lower *"average"* input quality. This adds another 3,000 samples, yielding *UltraMix-190k*, which remains 30% smaller than TuluDPO.

This adapted recipe captures a broader set of high-quality samples, especially for math and code, while also recovering strategically selected instruction following tasks by moderately relaxing quality constraints. As shown in Table 3, UltraMix-187k and UltraMix-190k contain a 5% and 10% increase in essential information seeking samples, as well as an increase in reasoning samples, relative to our previous mixture. We present additional details on our curation recipe in App. E.

Performance Analysis. Table 2 presents the corresponding evaluation results for UltraMix-187k and UltraMix-190k. Across nearly all benchmarks, the instruction-boosted UltraMix-190k significantly outperforms UltraMix-187k, and, most notably, TuluDPO.

Compared to UltraMix-170k, we observe substantial improvements for both new mixtures on IFEval, GSM8K, MATH, and coding, suggesting that the inclusion of more samples has increased overall performance across benchmarks. While UltraMix-187k still underperforms TuluDPO on IFEval and MATH, the additional 3,000 instruction following samples in UltraMix-190k have led to a compounding performance boost across all benchmarks despite a slightly lower input quality. This corroborates the observations by Lambert et al. [8], Djuhera et al. [13], and Cohere et al. [31] that instruction following samples can lead to noticeable cross-task performance improvements.

For Tulu3, coding performance for UltraMix-190k now closely approaches that of Code-Preference-Pairs, with HumanEval and HumanEval+ reaching 69.05% and 48.08%, respectively, compared to 70.68% and 50.61% (see Table 1). IFEval further improves to 81.13% from 80.35% on TuluDPO and from 79.38% on UltraMix-170k. In addition, GSM8K shows a notable increase to 82.48% from 79.48% on TuluDPO, and MATH rises to 23.56%, improving over TuluDPO's 22.66%.

For SmolLM, we observe similar trends for UltraMix-190k. GSM8K reaches 50.14% compared to 48.84% on TuluDPO and 47.96% on UltraMix-170k, while MATH improves to 6.05% from 5.63%. IFEval also increases to 52.54% from 51.97%. In addition, coding performance improves slightly to 2.44% and 1.83% on HumanEval and HumanEval+. However, these gains remain modest, as SmolLM lacks coding and reasoning capabilities due to its pre-training focus on conversational use.

Collectively, these results show that UltraMix-190k consistently achieves top-tier performance across a diverse set of tasks and models, offering substantial efficiency gains with 30% fewer samples than TuluDPO while exceeding its performance. We refer to App. F for a dedicated analysis of the corresponding DPO training efficiency. These findings confirm the effectiveness of our adapted reward-based data curation recipe. We designate UltraMix-190k as our final **UltraMix** DPO dataset.

6 Conclusion

In this work, we systematically analyzed and compared five widely used DPO post-training datasets: TuluDPO, UltraFeedback, ORPO, HelpSteer, and Code-Preference-Pairs. By annotating each preference pair with labels for input quality, difficulty, and preference reward, we identified high-reward, high-quality samples and developed a principled data curation strategy grounded in insights from targeted ablation studies. Our curation strategy led to the construction of UltraMix, a new DPO mixture that is 30% smaller than TuluDPO while achieving superior performance across a broad set of evaluation benchmarks. Our analysis also uncovered substantial inconsistencies in the ranking of chosen versus discarded completions in existing DPO datasets, suggesting that rankings and scores can be misleading when assessed by an independent reward model. The results presented in this study support several key takeaways: (a) high-reward samples are critical for driving performance gains in DPO training, (b) pre-trained reward models can offer more reliable signals than pre-annotated scores, and (c) the optimal composition of data mixtures is task-sensitive, requiring careful trade-offs between quality and task diversity, in particular for instruction following tasks. We validate the competitiveness of UltraMix by conducting extensive evaluations across diverse benchmarks and two LLMs of different sizes. By illustrating how quality-aware curation can reduce data requirements while increasing model performance, our work provides a strong foundation for future efforts in systematic, reward-aligned DPO dataset design. We discuss limitations and broader impact in App. G

Acknowledgments and Disclosure of Funding

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the research hub 6G-life (Grant 16KISK002), by the Bavarian Ministry of Science and the Arts and the Saxon Ministry for Science, Culture, and Tourism through the project Next Generation AI Computing (gAIn), by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the project 6G Future Lab Bavaria, and in part by IBM Research.

References

[1] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning Large Language Models with Human: A Survey. *arXiv preprint arXiv:2307.12966*, 2023.

[2] Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong. Large Language Model Alignment: A Survey. *arXiv preprint arXiv:2309.15025*, 2023.

[3] Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. LLM Post-Training: A Deep Dive into Reasoning Large Language Models. *arXiv preprint arXiv:2502.21321*, 2025.

[4] Zeyu Gan and Yong Liu. Towards a Theoretical Understanding of Synthetic Data in LLM Post-Training: A Reverse-Bottleneck Perspective. *arXiv preprint arXiv:2410.01720*, 2024.

[5] Guiyao Tie, Zeli Zhao, Dingjie Song, Fuyang Wei, Rong Zhou, Yurou Dai, Wen Yin, Zhejian Yang, Jiangyue Yan, Yao Su, Zhenhan Dai, Yifeng Xie, Yihan Cao, Lichao Sun, Pan Zhou, Lifang He, Hechang Chen, Yu Zhang, Qingsong Wen, Tianming Liu, Neil Zhenqiang Gong, Jiliang Tang, Caiming Xiong, Heng Ji, Philip S. Yu, and Jianfeng Gao. A Survey on Post-training of Large Language Models, 2025. URL https://arxiv.org/abs/2503.06072.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[7] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *Advances in Neural Information Processing Systems*, 2023.

[8] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing Frontiers in Open Language Model Post-Training, 2025. URL https://arxiv.org/abs/2411.15124.

[9] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting Language Models with Scaled AI Feedback, 2024. URL https://arxiv.org/abs/2310.01377.

[10] Maxime Labonne. ORPO-DPO-mix-40k. [[https://huggingface.co/datasets/mlabonne/orpo-dpo-mix-40k]]([https://huggingface.co/datasets/mlabonne/orpo-dpo-mix-40k]), 2024.

[11] Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. HelpSteer2-Preference: Complementing Ratings with Preferences, 2024. URL https://arxiv.org/abs/2410.01257.

[12] Vezora. Code-Preference-Pairs DPO Mix. [[https://huggingface.co/datasets/Vezora/Code-Preference-Pairs]), 2024.

[13] Aladin Djuhera, Swanand Ravindra Kadhe, Syed Zawad, Farhan Ahmed, Heiko Ludwig, and Holger Boche. Fixing It in Post: A Comparative Study of LLM Post-Training Data Quality and Model Performance, 2025. URL https://arxiv.org/abs/2506.06522.

[14] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment Data Synthesis from Scratch by Prompting Aligned LLMs with Nothing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Pnk7vMbznK.

[15] 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, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model, 2025. URL https://arxiv.org/abs/2502.02737.

[16] Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. OpenLLM Leaderboard V1. https://huggingface.co/docs/leaderboards/en/open_llm_leaderboard/archive, 2023.

[17] Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open-LLM Leaderboard V2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.

[18] Aaron Grattafiori et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

[19] Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. WildChat: 1M ChatGPT Interaction Logs in the Wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

[20] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. *arXiv preprint arXiv:2301.13688*, 2023.

[21] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian

Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. GPT-4o System Card, 2024. URL https://arxiv.org/abs/2410.21276.

[22] Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic Preference Optimization without Reference Model, 2024. URL https://arxiv.org/abs/2403.07691.

[23] Maxime Labonne. mlabonne/llm-datasets: Curated List of Datasets and Tools for Post-Training. GitHub repository, 2025. URL https://github.com/mlabonne/llm-datasets.

[24] Allen Institute for AI (AllenAI). Open Instruct: AllenAI's Post-Training Codebase. https://github.com/allenai/open-instruct, 2023.

[25] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The Language Model Evaluation Harness, 07 2024. URL https://zenodo.org/records/12608602.

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

[27] Llama Team. Meta Llama Guard 2. `https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md`, 2024.

[28] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF workflow: From reward modeling to online RLHF. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=a13aYUU9eU`.

[29] Anthropic. Dataset Card for HH-RLHF by Anthropic. `[[https://huggingface.co/datasets/Anthropic/hh-rlhf])`, 2024.

[30] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding Dataset Difficulty with $\mathcal{V}$-Usable Information. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR, 17–23 Jul 2022.

[31] Team Cohere, :, Aakanksha, Arash Ahmadian, Marwan Ahmed, Jay Alammar, Milad Alizadeh, Yazeed Alnumay, Sophia Althammer, Arkady Arkhangorodsky, Viraat Aryabumi, Dennis Aumiller, Raphaël Avalos, Zahara Aviv, Sammie Bae, Saurabh Baji, Alexandre Barbet, Max Bartolo, Björn Bebensee, Neeral Beladia, Walter Beller-Morales, Alexandre Bérard, Andrew Berneshawi, Anna Bialas, Phil Blunsom, Matt Bobkin, Adi Bongale, Sam Braun, Maxime Brunet, Samuel Cahyawijaya, David Cairuz, Jon Ander Campos, Cassie Cao, Kris Cao, Roman Castagné, Julián Cendrero, Leila Chan Currie, Yash Chandak, Diane Chang, Giannis Chatziveroglou, Hongyu Chen, Claire Cheng, Alexis Chevalier, Justin T. Chiu, Eugene Cho, Eugene Choi, Eujeong Choi, Tim Chung, Volkan Cirik, Ana Cismaru, Pierre Clavier, Henry Conklin, Lucas Crawhall-Stein, Devon Crouse, Andres Felipe Cruz-Salinas, Ben Cyrus, Daniel D'souza, Hugo Dalla-Torre, John Dang, William Darling, Omar Darwiche Domingues, Saurabh Dash, Antoine Debugne, Théo Dehaze, Shaan Desai, Joan Devassy, Rishit Dholakia, Kyle Duffy, Ali Edalati, Ace Eldeib, Abdullah Elkady, Sarah Elsharkawy, Irem Ergün, Beyza Ermis, Marzieh Fadaee, Boyu Fan, Lucas Fayoux, Yannis Flet-Berliac, Nick Frosst, Matthias Gallé, Wojciech Galuba, Utsav Garg, Matthieu Geist, Mohammad Gheshlaghi Azar, Ellen Gilsenan-McMahon, Seraphina Goldfarb-Tarrant, Tomas Goldsack, Aidan Gomez, Victor Machado Gonzaga, Nithya Govindarajan, Manoj Govindassamy, Nathan Grinsztajn, Nikolas Gritsch, Patrick Gu, Shangmin Guo, Kilian Haefeli, Rod Hajjar, Tim Hawes, Jingyi He, Sebastian Hofstätter, Sungjin Hong, Sara Hooker, Tom Hosking, Stephanie Howe, Eric Hu, Renjie Huang, Hemant Jain, Ritika Jain, Nick Jakobi, Madeline Jenkins, JJ Jordan, Dhruti Joshi, Jason Jung, Trushant Kalyanpur, Siddhartha Rao Kamalakara, Julia Kedrzycki, Gokce Keskin, Edward Kim, Joon Kim, Wei-Yin Ko, Tom Kocmi, Michael Kozakov, Wojciech Kryściński, Arnav Kumar Jain, Komal Kumar Teru, Sander Land, Michael Lasby, Olivia Lasche, Justin Lee, Patrick Lewis, Jeffrey Li, Jonathan Li, Hangyu Lin, Acyr Locatelli, Kevin Luong, Raymond Ma, Lukáš Mach, Marina Machado, Joanne Magbitang, Brenda Malacara Lopez, Aryan Mann, Kelly Marchisio, Olivia Markham, Alexandre Matton, Alex McKinney, Dominic McLoughlin, Jozef Mokry, Adrien Morisot, Autumn Moulder, Harry Moynehan, Maximilian Mozes, Vivek Muppalla, Lidiya Murakhovska, Hemangani Nagarajan, Alekhya Nandula, Hisham Nasir, Shauna Nehra, Josh Netto-Rosen, Daniel Ohashi, James Owers-Bardsley, Jason Ozuzu, Dennis Padilla, Gloria Park, Sam Passaglia, Jeremy Pekmez, Laura Penstone, Aleksandra Piktus, Case Ploeg, Andrew Poulton, Youran Qi, Shubha Raghvendra, Miguel Ramos, Ekagra Ranjan, Pierre Richemond, Cécile Robert-Michon, Aurélien Rodriguez, Sudip Roy, Sebastian Ruder, Laura Ruis, Louise

Rust, Anubhav Sachan, Alejandro Salamanca, Kailash Karthik Saravanakumar, Isha Satyakam, Alice Schoenauer Sebag, Priyanka Sen, Sholeh Sepehri, Preethi Seshadri, Ye Shen, Tom Sherborne, Sylvie Shang Shi, Sanal Shivaprasad, Vladyslav Shmyhlo, Anirudh Shrinivason, Inna Shteinbuk, Amir Shukayev, Mathieu Simard, Ella Snyder, Ava Spataru, Victoria Spooner, Trisha Starostina, Florian Strub, Yixuan Su, Jimin Sun, Dwarak Talupuru, Eugene Tarassov, Elena Tommasone, Jennifer Tracey, Billy Trend, Evren Tumer, Ahmet Üstün, Bharat Venkitesh, David Venuto, Pat Verga, Maxime Voisin, Alex Wang, Donglu Wang, Shijian Wang, Edmond Wen, Naomi White, Jesse Willman, Marysia Winkels, Chen Xia, Jessica Xie, Minjie Xu, Bowen Yang, Tan Yi-Chern, Ivan Zhang, Zhenyu Zhao, and Zhoujie Zhao. Command A: An Enterprise-Ready Large Language Model, 2025. URL https://arxiv.org/abs/2504.00698.

[32] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

[33] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, Eric Chung, and Chris Alexiuk. Llama-Nemotron: Efficient Reasoning Models, 2025. URL https://arxiv.org/abs/2505.00949.

[34] NovaSky Team. Sky-T1: Train your own O1 preview model within $450. https://novasky-ai.github.io/posts/sky-t1, 2025.

[35] Bespoke Labs. Bespoke-Stratos: The Unreasonable Effectiveness of Reasoning Distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025.

[36] Sathwik Tejaswi Madhusudhan, Shruthan Radhakrishna, Jash Mehta, and Toby Liang. Millions Scale Dataset Distilled from R1-32B. https://huggingface.co/datasets/ServiceNow-AI/R1-Distill-SFT, 2025.

[37] Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Imitate, Explore, and Self-Improve: A Reproduction Report on Slow-thinking Reasoning Systems, 2024. URL https://arxiv.org/abs/2412.09413.

[38] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[39] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[40] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. *arXiv preprint arXiv:2210.09261*, 2022.

[41] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *ArXiv*, 2018.

[42] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[43] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *arXiv preprint arXiv:1907.10641*, 2019.

[44] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*, 2023.

[45] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021.

[46] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *Advances in Neural Information Processing Systems*, 2021.

A Post-Training Large Language Models

While *pre-training* enables models to acquire broad linguistic competence and world knowledge, *post-training* adapts models to follow instructions, specialize in tasks, and behave helpfully.

A.1 General Post-Training Methods

Post-training typically involves instruction tuning through supervised fine-tuning (SFT), followed by preference fine-tuning, for example using reinforcement learning (RL).

Supervised Fine-Tuning (SFT). SFT adapts pre-trained LLMs for either broad or specialized domain knowledge. This is done using instruction-response examples, which may originate from human annotations or synthetic generation. Through next-token prediction, the model learns to generalize across diverse domains and specialized tasks, resulting in an instruction-tuned model that responds helpfully. Although SFT enhances a model's ability to follow instructions and handle a wide range of tasks, it does not necessarily align the model's outputs with pre-defined preferences.

Preference Fine-Tuning. The goal of preference fine-tuning is is to align the SFT model's output distribution with behavior that appeals to human preferences or adheres to task-specific objectives. This is commonly achieved by leveraging a learned reward signal or explicit preference annotations to steer the model toward generating responses that reflect pre-defined behavior such as increased helpfulness, harmlessness, honesty, and style. Widely used methods for preference-based alignment include Proximal Policy Optimization (PPO) [6] and Direct Preference Optimization (DPO) [7].

Reasoning Alignment. Recent work has explored RL methods that enhance a model's capacity for structured reasoning. For instance, Group Relative Policy Optimization (GRPO) [32] has been proposed to elicit deeper, multi-step thinking by refining reward signals through group-based preference aggregation. In parallel, several reasoning-centric datasets have been introduced [33–37], focusing on task formats that explicitly promote step-by-step problem solving and reflective thought processes.

A.2 Focus on DPO

The primary goal of this paper is to analyze the quality and composition of preference optimization datasets while keeping the overall training procedure fixed, enabling a fair comparison. In particular, we focus on DPO, which has become the most widely adopted approach for preference-based alignment due to its simplicity and efficiency. Unlike PPO that requires both reward modeling and costly policy rollouts, DPO operates directly on preference pairs, making it well suited for dataset-centric comparisons given the large availability of several open-source DPO corpora.

Similar efforts have been pursued by Djuhera et al. [13] for post-training with SFT, demonstrating that task-aware dataset design can improve performance while reducing overall dataset size. However, the diversity of existing post-training methods, including SFT, PPO, and others, implies that dataset requirements are highly method-dependent, complicating direct comparisons across approaches. For instance, unlike SFT, DPO is based on the Bradley-Terry formulation, which relies on pairwise preference samples for optimization. As a result, curation strategies developed for SFT may not be directly applicable to DPO datasets. To address this gap, we focus specifically on post-training with DPO, analyzing the role of preference rewards and assessing the reliability of pre-annotated scores, thus going beyond binary preference labels to enable more insightful dataset construction.

Our evaluation demonstrates that reward-based dataset design is a critical factor in DPO training: by systematically annotating, analyzing, and curating open-source DPO corpora based on preference rewards and quality, we construct UltraMix, a leaner mixture that outperforms larger individual datasets across diverse benchmarks.

B Experimental Setup

We present details on the training and evaluation configurations used throughout our experiments.

B.1 DPO Fine-Tuning

To ensure reproducibility and comparability, we fine-tune all models for DPO using AllenAI's *Open-Instruct* framework[2]. Unlike SFT, where loss aggregation across tokens requires careful handling to avoid length bias, the DPO objective is inherently length-normalized [7, 8]. This formulation ensures that completions of different lengths are treated equitably, and it avoids instabilities from batch composition or sequence length distributions observed in prior SFT pipelines.

To ensure consistency, we fix the training hyperparameters for each model across all experiments and apply DPO strictly on models that have already been fine-tuned via SFT using their instruction-tuning datasets. To this end, we use the SFT checkpoints from HuggingFace. For Tulu3-8B-SFT, we adopt the same hyperparameters as in Lambert et al. [8], and for SmolLM2-1.7B-SFT, we follow Allal et al. [15]. Specifically, we set the KL-penalty coefficient (referred to as `dpo_beta` in *Open-Instruct*) to 5 for Tulu3-8B-SFT and to 0.5 for SmolLM2-1.7B-SFT. We use the length-normalized DPO loss (`dpo_loss_type=norm`), following the recommendation in Lambert et al. [8].

Training is performed using BF16 mixed precision with Fully Sharded Data Parallelism (FSDP) on 8 × NVIDIA A100 80GB GPUs. Table 4 provides the DPO training configurations for both models.

Table 4: DPO training hyperparameters for the Tulu3-8B-SFT and SmolLM2-1.7B-SFT models.

Parameter	Tulu3-8B-SFT	SmolLM2-1.7B-SFT
Total Batch Size	128	128
Per-Device Batch Size	1	1
Gradient Accumulation Steps	16	16
Max Sequence Length	2048	1024
Number of Epochs	1	2
Learning Rate	5×10^{-7}	1×10^{-6}
LR Scheduler	Linear	Linear
Warmup Ratio	0.1	0.0
Weight Decay	0.0	0.0
DPO Beta	5	0.5

B.2 Evaluation Setup

We assess model performance using the *LM Evaluation Harness* framework [25], a widely adopted standard for evaluating language models across diverse benchmark suites. To ensure a comprehensive and task-diverse evaluation, we include benchmarks from *Open LLM Leaderboards* V1 [16] and V2 [17] to gauge downstream task performance under competitive public standards. This spans *Knowledge* (e.g., MMLU [38], TruthfulQA [39]), *Reasoning* (e.g., BBH [40], ARC-C [41]), *Commonsense Understanding* (e.g., HellaSwag [42], WinoGrande [43]), *Instruction Following* (e.g., IF-Eval [44]), *Mathematical Reasoning* (e.g., GSM8K [45], MATH [46]), and *Coding* (e.g., HumanEval, HumanEval+ [26]), ensuring a fair comparison between models and data mixtures.

[2] `https://github.com/allenai/open-instruct`

C Magpie Annotations

This section provides a brief overview of the Magpie annotation framework.

C.1 General Overview

Magpie [14] is a *self-synthesis* framework that derives alignment-oriented annotations from open-weight, instruction-tuned language models without requiring human labels or handcrafted seed prompts. Although the framework is also capable of generating new instruction-response pairs, in this work we rely exclusively on its *annotation* capabilities.

Magpie employs specialized judge models to automatically label individual samples across multiple dimensions (e.g., prompt quality, task type, and safety), making it possible to obtain large-scale annotations that would be prohibitively expensive to collect through manual labeling. This additional metadata enables more principled dataset filtering, stratification, and targeted analysis.

Magpie supports the following annotation labels:

- **Input Quality** (*very poor – excellent*): assesses prompt clarity, specificity, and structure, accompanied by a short textual justification (*"quality explanation"*).
- **Task Category**: maps each prompt to one of 12 categories, such as *Coding & Debugging, Reasoning, Information Seeking, Brainstorming, Creative Writing, Advice Seeking, Math, Planning, Editing, Role Playing, Data Analysis*, or *Other*.
- **Input Difficulty** (*very easy – very hard*): estimates the cognitive demand of the prompt. Each sample is further tagged with *intent* (user goal) and *knowledge* (required competence).
- **Response Quality**: provides a scalar judgment of the response, scored by a reward model.
- **Safety**: classified using a dedicated safety model.
- **Language**: detects the language of the prompt.

A key feature of Magpie is its modularity: any instruction-tuned LLM can, in principle, serve as the judge model. In its default configuration, Magpie relies on Llama-3-8B-Instruct [18] for general annotation and Llama-Guard 2 [27] for safety assessment. In our pipeline, we use Llama-3.3-70B-Instruct [18] as the judge model given its demonstrated reliability, particularly for quality and difficulty annotations, as assessed by Djuhera et al. [13]. In addition, we also incorporate the error-tolerant JSON parsing extensions to Magpie from Djuhera et al. [13], which help handle formatting inconsistencies and free-form outputs during the annotation process.

C.2 Preference Reward Assessment

We assess preference rewards by adapting Magpie to evaluate the response quality of each completion. To this end, we rely on an independent, specialized reward model that has been fine-tuned on multiple preference datasets. Specifically, we use the Llama-3-8B-Instruct-based reward model introduced by Dong et al. [28], which is trained using the Bradley-Terry formulation on a diverse set of high-quality preference datasets, including HH-RLHF [29] and SHP [30]. This reward model has demonstrated strong alignment with human preference judgments, making it a reliable signal for quantifying how well a model completion satisfies the given instruction.

The model assigns a scalar reward score (higher is better) to both the chosen and rejected completions in each preference pair. We use these scores in our analysis to verify the correctness of the original preference order. Implementation details can be found in our publicly released Magpie codebase, and we make all preference reward scores available as part of our annotated datasets.

D Extended Analysis

We present extended analyses of our annotated DPO post-training datasets, covering dataset composition, task distributions, difficulty and quality metrics, as well as preference reward evaluations.

D.1 Dataset Compositions per Task Category

Tables 5 and 6 compare the composition of each DPO dataset by task category, as labeled by Magpie. This provides a *sample-level* view of how different task types are distributed across all examined open-source datasets, as well as our curated DPO mixtures.

The breakdown in Table 5 complements the analysis presented in Fig. 1, showing that general-purpose datasets like TuluDPO and ORPO are more strategically distributed with high proportions in information seeking, math, and coding.

Table 6 compares the distributions for each DPO mixture according to our applied curation recipes, showing that UltraMix-187k and UltraMix-190k increase the proportion of information seeking and reasoning samples, leading to compounding improvements across benchmarks as a result of both a broader task mix associated with instruction following and an increase in absolute sample count.

Table 5: Dataset composition per task category for all examined open-source DPO datasets.

Task Category	TuluDPO	UltraFeedback	ORPO	HelpSteer	CodePref
Information seeking	38.3%	49.1%	37.6%	50.8%	0.0%
Reasoning	9.1%	9.8%	4.7%	3.4%	0.0%
Coding & Debugging	12.4%	13.4%	8.5%	5.2%	100.0%
Editing	1.6%	2.4%	1.2%	2.8%	0.0%
Math	16.7%	3.6%	28.9%	3.3%	0.0%
Advice seeking	3.1%	3.9%	4.4%	6.8%	0.0%
Planning	1.9%	3.4%	3.5%	6.0%	0.0%
Creative writing	9.8%	6.3%	4.9%	8.6%	0.0%
Brainstorming	2.1%	3.0%	2.4%	5.4%	0.0%
Data analysis	2.0%	2.1%	1.5%	1.7%	0.0%
Role playing	1.5%	1.5%	1.4%	5.1%	0.0%
Others	1.5%	1.3%	1.0%	0.8%	0.0%

Table 6: Dataset composition per task category for our curated UltraMix variants.

Task Category	UltraMix-170k	UltraMix-187k	UltraMix-190k (**UltraMix**)
Information seeking	31.4%	31.8%	32.7%
Reasoning	5.6%	7.1%	7.4%
Coding & Debugging	21.5%	21.0%	20.7%
Editing	1.6%	1.5%	1.5%
Math	18.6%	19.3%	19.0%
Advice seeking	2.9%	2.6%	2.5%
Planning	3.1%	2.4%	2.3%
Creative writing	9.7%	8.9%	8.8%
Brainstorming	1.7%	1.5%	1.5%
Data analysis	2.7%	2.4%	2.4%
Role playing	0.9%	1.1%	1.1%
Others	0.2%	0.2%	0.1%

D.2 Query Difficulty per Task Category

Fig. 6 to Fig. 10 show the distribution of query difficulty across task categories for each DPO dataset. This analysis complements Fig. 2a, confirming that most prompts are labeled as *"hard"* or *"medium"*, with only a smaller fraction categorized as *"easy"*. This suggests that preference tuning, across both general-purpose and domain-specific datasets, emphasizes more challenging instruction-response pairs, which may contribute to stronger alignment and improved downstream performance.

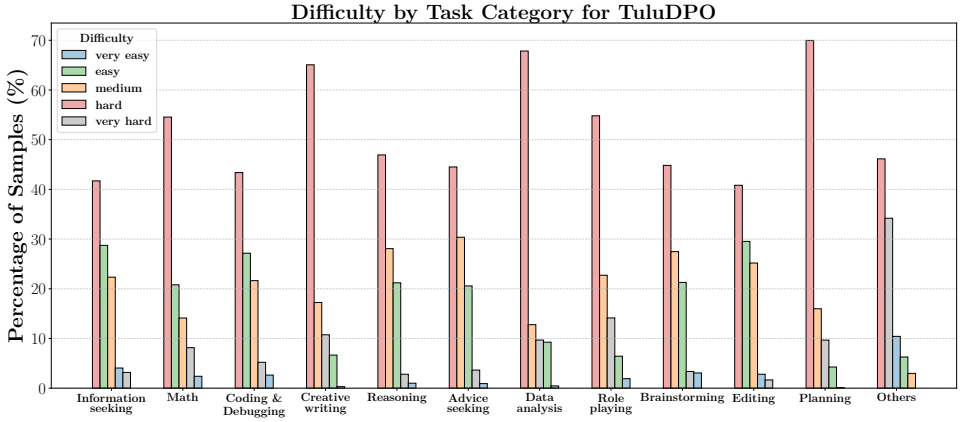


Figure 6: Distribution of query difficulty per task category for TuluDPO.

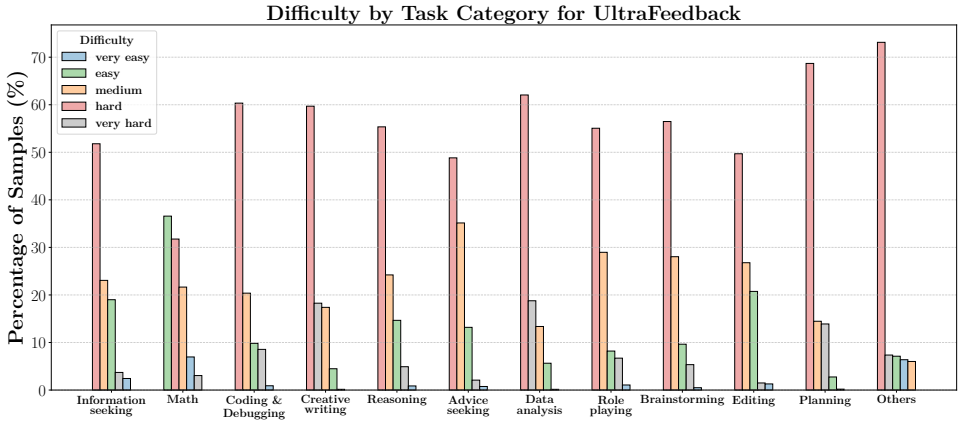


Figure 7: Distribution of query difficulty per task category for UltraFeedback.

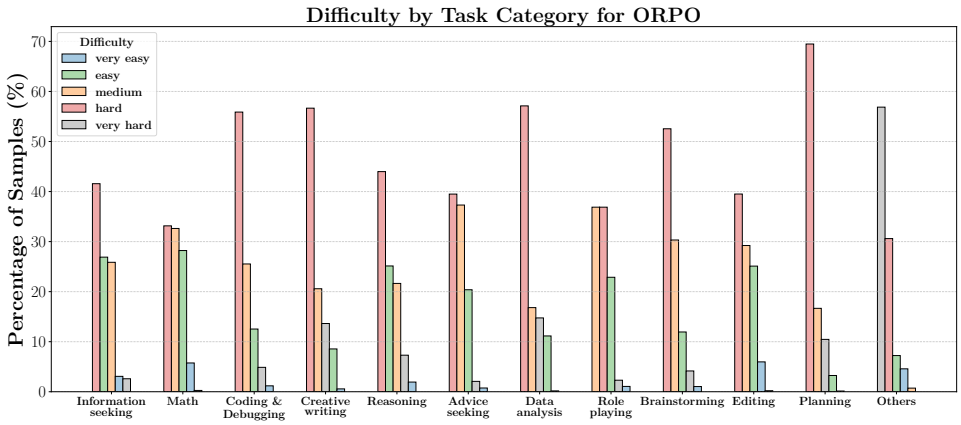


Figure 8: Distribution of query difficulty per task category for ORPO.

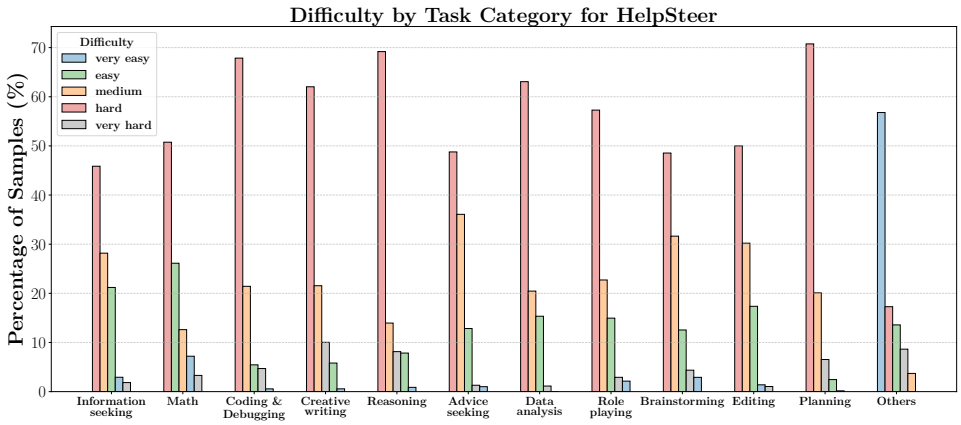


Figure 9: Distribution of query difficulty per task category for HelpSteer.

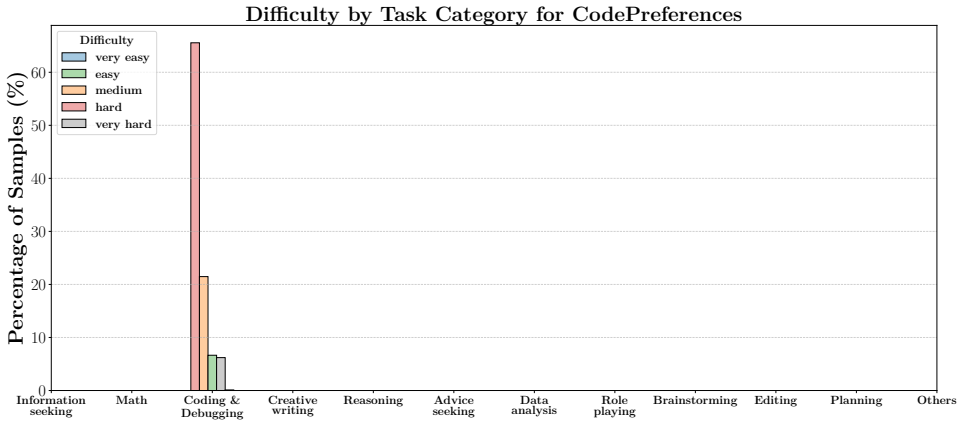


Figure 10: Distribution of query difficulty per task category for Code-Preference-Pairs.

D.3 Input Quality per Task Category

Fig. 11 to Fig. 15 show the distribution of input quality across task categories for each DPO dataset. This analysis complements Fig. 2b, confirming that ORPO, and TuluDPO, UltraFeedback, ORPO, and Code-Preference-Pairs contain predominantly high-quality prompts, with more than 75% rated as either *"good"* or *"excellent"*. In contrast, HelpSteer exhibits a more even distribution, with a non-negligible portion of prompts rated as *"average"*, *"poor"*, or *"very poor"*. Many of these samples show a lack of context or underspecified user intent, suggesting that a quality filter could be applied to remove low-quality instances.

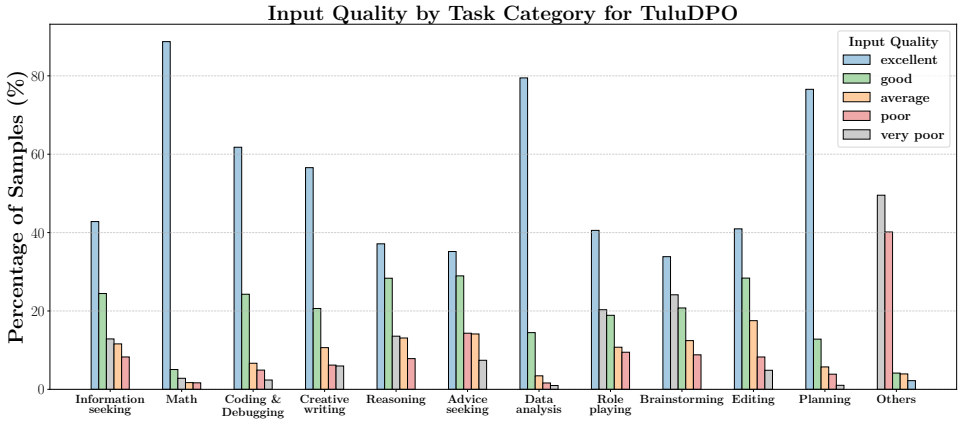


Figure 11: Distribution of input quality per task category for TuluDPO.

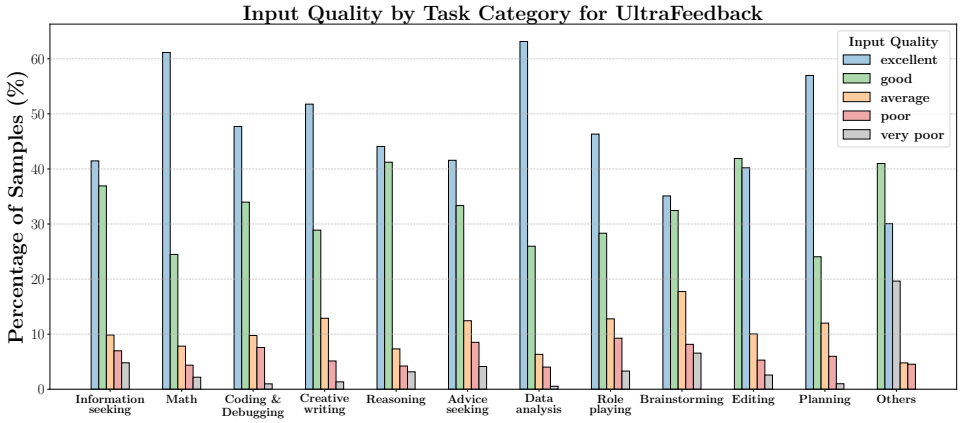


Figure 12: Distribution of input quality per task category for UltraFeedback.

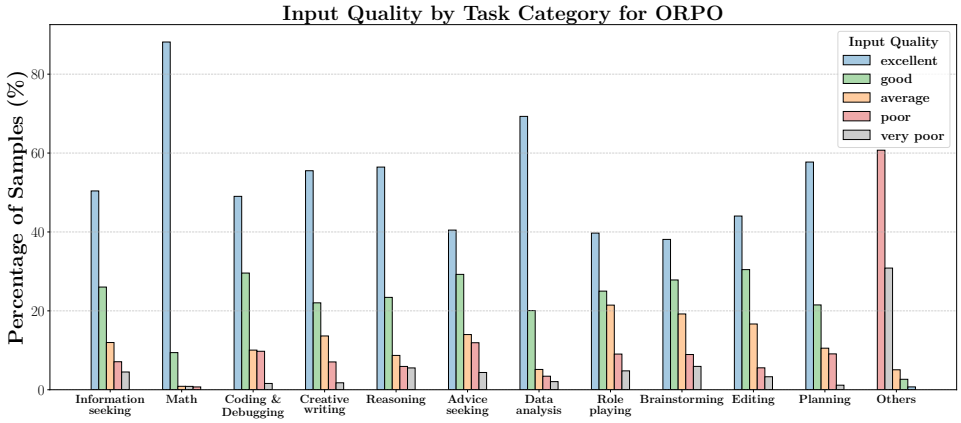


Figure 13: Distribution of input quality per task category for ORPO.

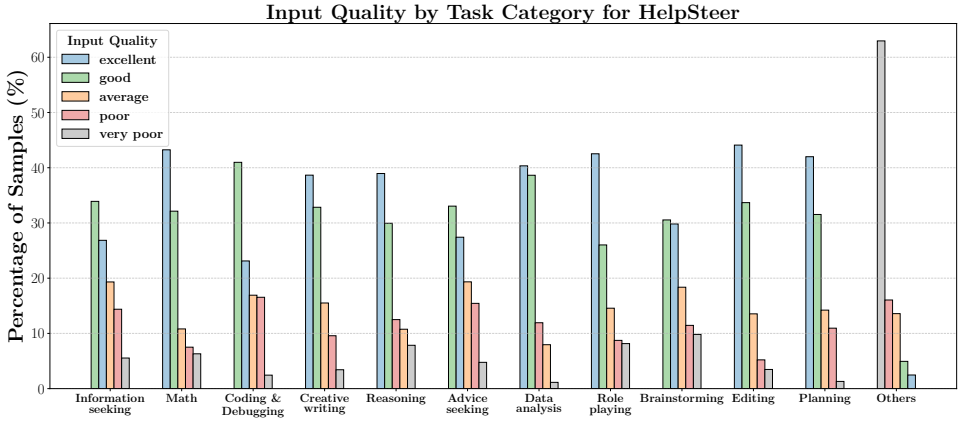


Figure 14: Distribution of input quality per task category for HelpSteer.

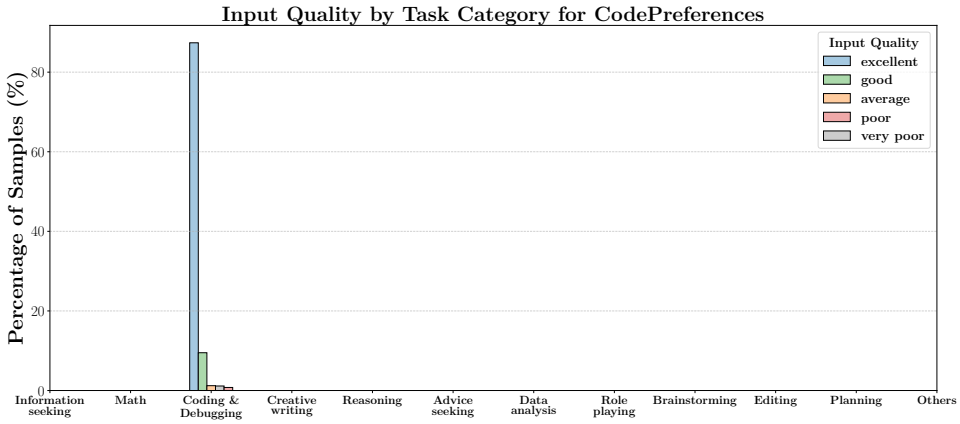


Figure 15: Distribution of input quality per task category for Code-Preference-Pairs.

D.4 Preference Reward Analysis

We present additional results and insights on the distribution of preference rewards across task categories, as well as their relationship to query difficulty and input quality.

D.4.1 Preference Rewards per Task Category

Fig. 16 shows the distribution of preference rewards per DPO dataset across task categories for all samples in which the chosen completion is correctly preferred over the rejected one, according to our reward model. This analysis complements Fig. 3 and confirms that most categories contain a non-negligible portion of misaligned rewards in preference pairs. This suggests that pre-annotated scores and binary preference orders do not always align with the judgment of an independent, reward-aligned model. These findings further support the use of reward-based filtering to identify high-reward samples that may enhance alignment during DPO while simultaneously reducing dataset size by filtering out noisy or redundant examples.

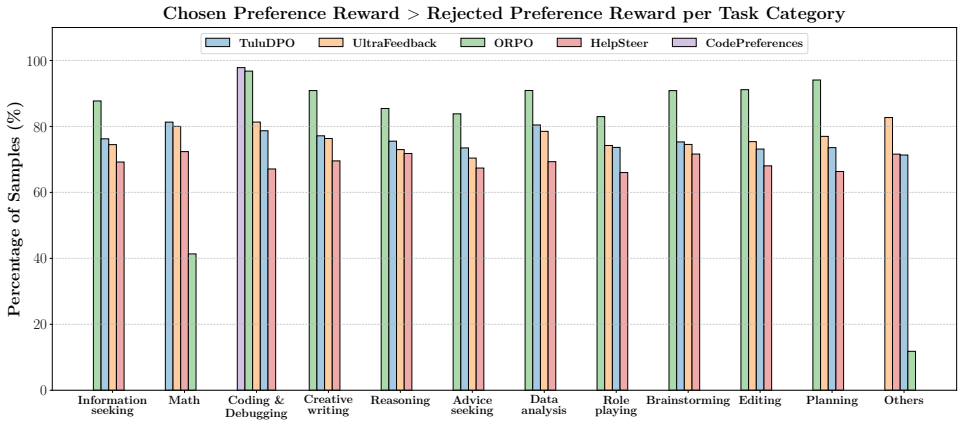


Figure 16: Distribution of preference rewards across task categories for all examined open-source DPO datasets. We present the reward-aligned proportions in which the chosen completions are correctly preferred over the rejected ones, according to our reward model.

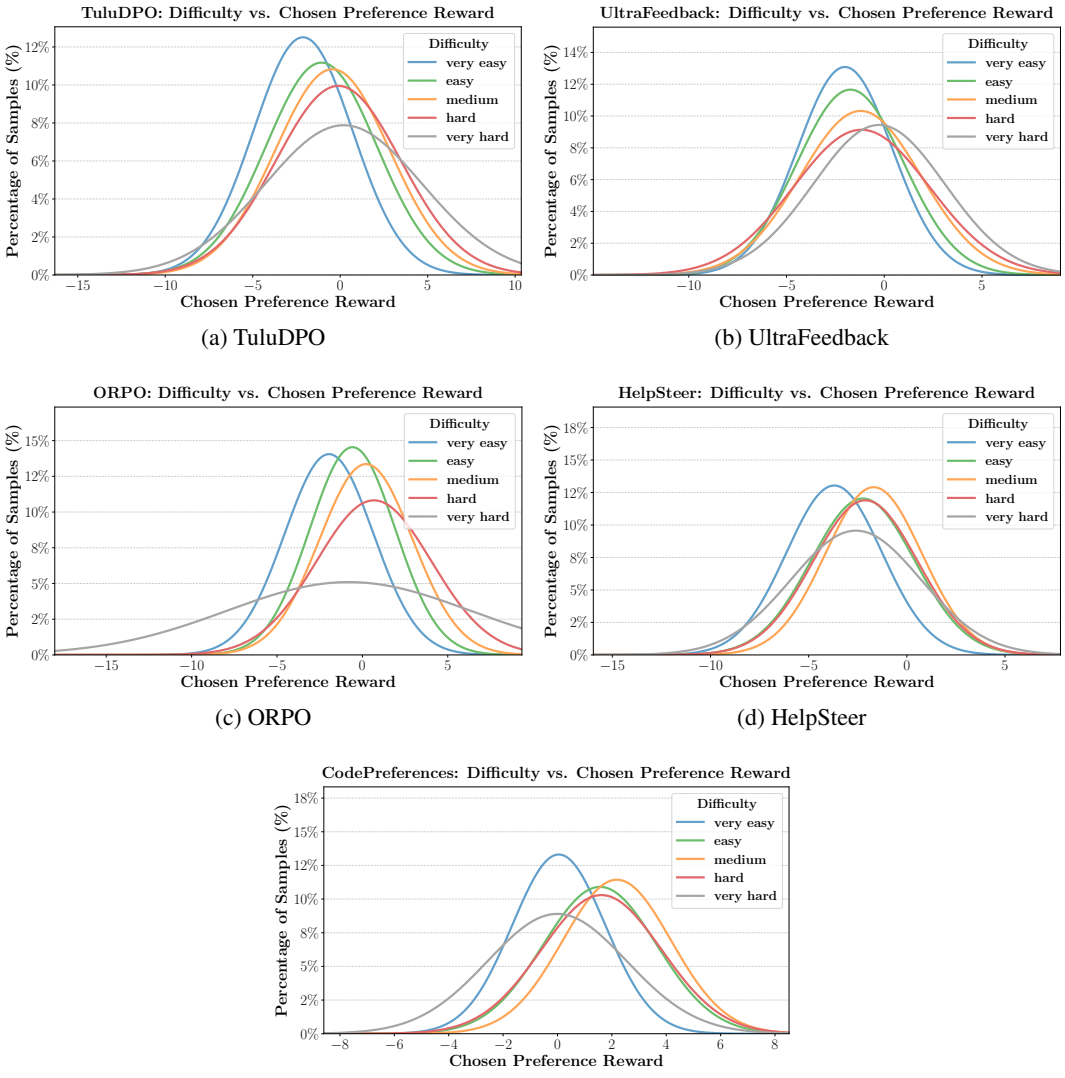
D.4.2 Preference Reward vs. Query Difficulty

Fig. 17 compares average preference rewards for chosen and rejected completions (filtered and aligned with our reward model) across different levels of query difficulty. We observe that, for both chosen and discarded responses, reward scores tend to increase as query difficulty increases. This extends our previous findings, suggesting that higher-difficulty prompts are more likely to be associated with high-reward completions. The trend is most prominent in Fig. 17b, while Fig. 17a shows Code-Preference-Pairs as a notable outlier. As discussed in the main body and observed in Fig. 10 and Fig. 16, this skew arises from the dataset's unique composition, often featuring code samples where the only difference between completions is the presence or absence of inline comments, thus resulting in a disproportionate distribution of preference rewards across difficulty levels.

In addition, Fig. 18 shows the distribution of average chosen preference rewards across query difficulty levels for each DPO dataset individually, providing further insights into per-dataset-level characteristics. Specifically, Fig. 18a to Fig. 18e confirm our above statements, showing that more difficult samples tend to yield higher preference reward scores as observed by the visible shifts in chosen preference rewards for increasing difficulty across datasets.

(a) Difficulty vs. average chosen preference reward. (b) Difficulty vs. average rejected preference reward.

Figure 17: Comparison of average preference rewards against query difficulty.



(a) TuluDPO (b) UltraFeedback

(c) ORPO (d) HelpSteer

(e) Code-Preference-Pairs

Figure 18: Comparison of chosen preference reward distributions against query difficulty for all examined open-source DPO datasets.

D.4.3 Preference Reward vs. Input Quality

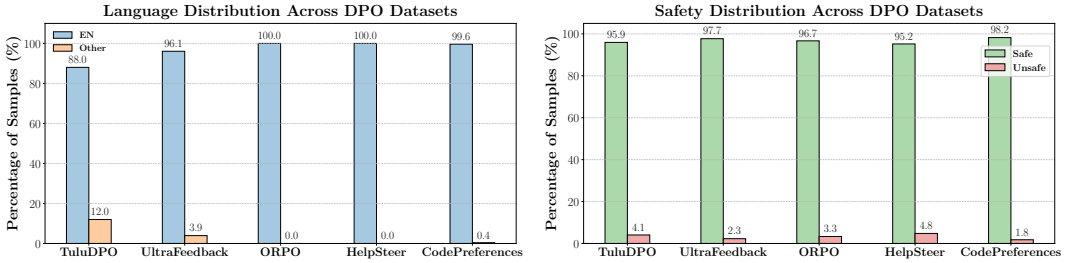
Fig. 19 compares average preference rewards for chosen and rejected completions (filtered and aligned with our reward model) across different levels of input quality. We observe that, similarly to query difficulty, reward scores tend to increase as input quality increases. This further corroborates our previous statements, suggesting that high-quality instructions are likely to be associated with high-reward completions and are thus an important factor for effective DPO alignment. We provide additional per-dataset-level distributions in Fig. 21 for all our examined open-source DPO datasets.

(a) Input quality vs. average chosen preference reward. (b) Input quality vs. average rejected preference reward.

Figure 19: Comparison of average preference rewards against input quality.

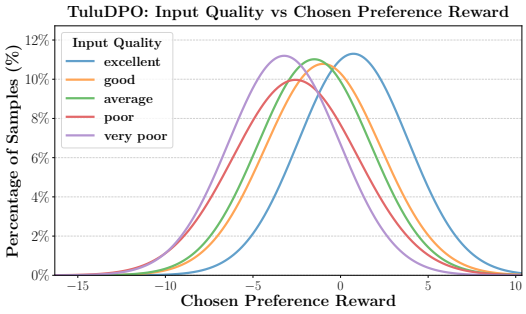
D.5 Language and Safety Analysis

Fig. 20 shows the distribution of language and safety across all examined open-source DPO datasets. We find that most datasets are predominantly English (96–99%), with the exception of TuluDPO, which includes approximately 12% non-English samples, primarily in French and German. All datasets are also rated as overwhelmingly safe (95–98%). As already discussed by Djuhera et al. [13], language and safety are not significantly correlated with post-training performance. For this reason, we do not include these attributes in our data curation recipes. Nevertheless, we release all language and safety labels as part of our annotated datasets.

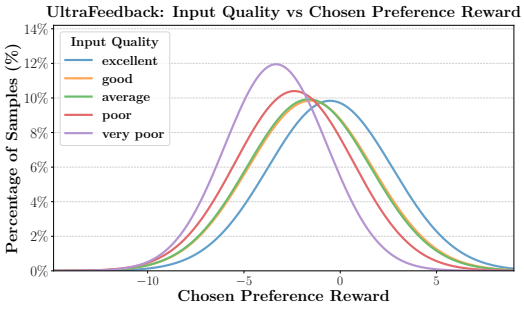


(a) Distribution of language labels per dataset. (b) Distribution of safety labels per dataset.

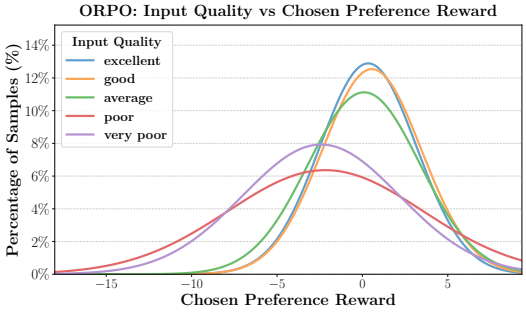
Figure 20: Distribution of language and safety labels for all examined open-source DPO datasets.



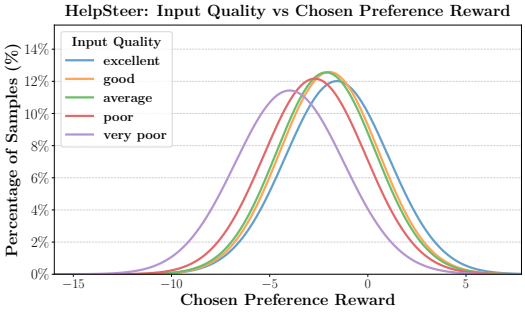
(a) TuluDPO



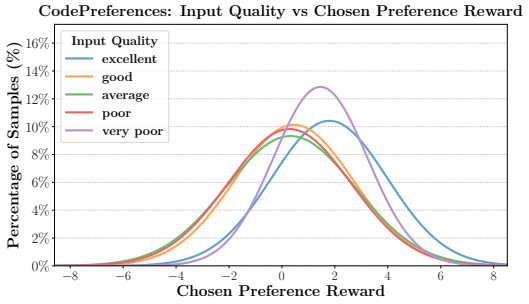
(b) UltraFeedback



(c) ORPO



(d) HelpSteer



(e) Code-Preference-Pairs

Figure 21: Comparison of chosen preference reward distributions against input quality for all examined open-source DPO datasets.

E Data Curation Recipe Details

This section details the quality-, reward-, and task-based data curation procedure used to construct our *UltraMix* DPO mixture. The general steps of the algorithm are summarized in Fig. 22.

In **Step 1**, we filter each dataset to retain only samples that satisfy three basic conditions:

- First: input quality is rated at least *"good"*.
- Second: difficulty is above *"very easy"*.
- Third: the chosen response achieves a higher preference reward than the discarded one.

This ensures that only meaningful preference pairs with reliable alignment signals are considered.

Step 2 applies reward-based thresholds:

- For TuluDPO, UltraFeedback, ORPO, and HelpSteer, we retain samples with preference rewards above the 25th percentile of each dataset.
- For Code-Preference-Pairs, which is substantially larger and only contains code samples, we apply a stricter cutoff at the 80th percentile to avoid over-representing code tasks.

This step prioritizes high-reward, high-quality pairs while maintaining dataset diversity.

Step 3 performs deduplication across sources. Since TuluDPO includes significant overlap with UltraFeedback, we remove duplicate prompts to prevent overweighting repeated instructions. Despite the two datasets containing different completions for identical prompts, we justify deduplication as differences are generally minor and mostly stylistic, likely due to the use of similar LLM families during response generation. Deduplication eventually yields our initial *UltraMix-170k* mixture.

Step 4 augments underrepresented domains to restore task diversity. We first add back 25% more high-reward samples from all five datasets, yielding *UltraMix-187k* after deduplication. Next, to specifically strengthen instruction following performance, we selectively reintroduce additional information seeking and reasoning samples above the 70th percentile reward threshold, even when their input quality is slightly below *good*. After deduplication, this adds 3,000 instruction following samples, resulting in *UltraMix-190k*.

Together, these steps yield our final mixture, *UltraMix-190k* (referred to as *UltraMix*), a lean, high-quality mixture that is both reward-aligned and task-balanced. As shown in Table 2, *UltraMix* outperforms the larger TuluDPO baseline across multiple benchmarks while containing 30% fewer samples, validating the effectiveness of our curation strategy. Table 7 presents the composition of all UltraMix variants in terms of their source DPO datasets, showing that the majority of *UltraMix* samples (81%) originate from the high-performing TuluDPO dataset. This makes our curation process a rigorous, reward- and quality-based filtering of TuluDPO.

Table 7: Composition of UltraMix variants, shown as percentages of their source DPO datasets.

Source DPO Dataset	UltraMix-170k	UltraMix-187k	UltraMix-190k (**UltraMix**)
TuluDPO	68.02%	76.17%	81.13%
UltraFeedback	15.00%	9.20%	4.42%
ORPO	9.01%	5.47%	5.46%
HelpSteer	1.96%	1.63%	1.74%
Code-Preference-Pairs	6.02%	7.52%	7.25%

Quality-, Reward- and Task-Based Curation Recipe

Input: Annotated datasets $\{\mathcal{D}_k\}_{k=1}^{5}$ with Magpie labels for input quality (`input_quality`), difficulty (`difficulty`), preference rewards (`reward_chosen`, `reward_rejected`), and task category (`task_category`).

In addition: per-dataset reward quantiles $\{q_k\}$ with a dedicated $q_{\texttt{code}}$ for the code-only corpus, instruction following categories $\mathcal{IF}$ (e.g., information seeking, reasoning), under-representation

tolerance $\tau \in (0, 1)$, balancing thresholds q_* (primary) and r_{avg} (fallback).

Output: Curated set $\mathcal{D}_c$ that is high-quality, reward-aligned, and task-diverse.

Recipe:

1. **Initial Quality, Difficulty, and Reward Filter**: Build a candidate pool $\mathcal{P}$ of samples where

   $\mathcal{P} = \big\{S \in \cup_k \mathcal{D}_k \,\big|\, S[\texttt{input\_quality}] \in \{\texttt{good}, \texttt{excellent}\} \wedge S[\texttt{difficulty}] > \texttt{very easy} \wedge S[\texttt{reward\_chosen}] > S[\texttt{reward\_rejected}]\big\}$.

2. **Reward Thresholding**: For each dataset k, compute $P_{q_k}^{(k)}$ as the q_k-th percentile of $\{S[\texttt{reward\_chosen}] : S \in \mathcal{P} \cap \mathcal{D}_k\}$. Initialize

   $$\mathcal{D}_c \leftarrow \left\{S \in \mathcal{P} \cap \mathcal{D}_k \;\middle|\; S[\texttt{reward\_chosen}] \geq \begin{cases} P_{q_k}^{(k)}, & k \neq \texttt{code}, \\ P_{q_{\texttt{code}}}^{(\texttt{code})}, & k = \texttt{code} \end{cases}\right\}.$$

   Note: Raising/lowering q_k (or $q_{\texttt{code}}$) globally tightens/loosens the mixture; no absolute sample counts are required.

3. **Task Coverage Check**: Let $\pi_{\mathcal{D}}(c)$ and $\pi_{\mathcal{D}_c}(c)$ denote the fraction of samples in task c for the full union $\mathcal{D} = \cup_k \mathcal{D}_k$ and the current $\mathcal{D}_c$, respectively. Define the under-represented set

   $$\mathcal{C}_{\downarrow} = \Big\{ c \;\Big|\; \pi_{\mathcal{D}_c}(c) < (1 - \tau)\, \pi_{\mathcal{D}}(c) \Big\}.$$

4. **Task Boosting (for Instruction Following)**: For each $c \in \mathcal{C}_{\downarrow} \cap \mathcal{IF}$

   (a) form the residual pool $\mathcal{R}_c = \{S \in \mathcal{P} \setminus \mathcal{D}_c \mid S[\texttt{task\_category}] = c\}$.

   (b) Compute the category-specific high-reward cutoff $P_{q_*}^{(c)}$ over $\{S[\texttt{reward\_chosen}] : S \in \mathcal{R}_c,\ S[\texttt{input\_quality}] \in \{\texttt{good}, \texttt{excellent}\}\}$.

   (c) Add $\mathcal{D}_c \leftarrow \mathcal{D}_c \cup \big\{S \in \mathcal{R}_c \,\big|\, S[\texttt{input\_quality}] \in \{\texttt{good}, \texttt{excellent}\},\ S[\texttt{reward\_chosen}] \geq P_{q_*}^{(c)}\big\}$.

   Fallback (Quality Relaxation): If this set is empty, recompute a looser cutoff $P_{r_{\text{avg}}}^{(c)}$ over *average*-quality samples in $\mathcal{R}_c$ and add

   $$\big\{S \in \mathcal{R}_c \,\big|\, S[\texttt{input\_quality}] = \texttt{average},\ S[\texttt{reward\_chosen}] \geq P_{r_{\text{avg}}}^{(c)}\big\}.$$

   Repeat until $\pi_{\mathcal{D}_c}(c) \geq (1 - \tau)\pi_{\mathcal{D}}(c)$ or $\mathcal{R}_c$ is exhausted.

5. **Deduplication**: Compute a hash $h(S)$ over the prompts. For any duplicate hash h, keep only the instance with the highest $S[\texttt{reward\_chosen}]$.

Figure 22: Data curation algorithm for quality-, reward- and task-based DPO mixtures. Steps 1–2 apply margin and per-dataset reward quantiles (with a separate code threshold), Step 3 detects task under-representation, Step 4 restores instruction following coverage using a primary threshold q_* and a quality-relaxed fallback r_{avg}, and Step 5 deduplicates by prompt identity.

F Efficiency Gains

To assess efficiency and training cost, we report the number of processed tokens (computed with each model's distinct tokenizer), estimates for training FLOPs, and total GPU hours (on an 8xA100 cluster) in Table 8 for DPO training of Tulu3 and and SmolLM SFT models on TuluDPO and UltraMix. We find that the reduction in dataset size translates approximately linearly into efficiency gains, such that a 30% reduction in data for UltraMix results in a proportionate reduction in the number of processed tokens, FLOPs, and total GPU hours, validating the efficiency of our curation dataset.

Table 8: Efficiency comparison of DPO training for the Tulu3-8B-SFT and SmolLM2-1.7B-SFT models on TuluDPO vs. UltraMix. We report processed tokens (per tokenizer), estimated ExaFLOPs, and GPU hours (rounded to the nearest hour, excluding the initial warmup phase).

	Tulu3-8B-SFT		**SmolLM2-1.7B-SFT**	
Efficiency Metric	TuluDPO	UltraMix	TuluDPO	UltraMix
Tokens (↓)	120M	90M	70M	50M
ExaFLOPs (↓)	5.76	4.32	3.36	2.40
GPU Hours (↓)	11	8	10	7

G Limitations and Broader Impact

Limitations. While our study provides a comprehensive and principled analysis of open-source DPO datasets, several limitations remain. First, our annotations rely on the Magpie framework with LLM-as-a-judge prompting to evaluate input quality, query difficulty, and task type, complemented by a reward-model-based preference score to validate chosen completions. Although this combination allows scalable and fine-grained inspection of preference pairs, the subjectivity inherent in LLM-based judgments and the biases of the underlying reward model may affect label accuracy. Future improvements in judge models or reward assessment may thus shift the results. Second, our focus is exclusively on DPO due to its popularity and the availability of large open-source corpora. Other alignment methods such as PPO, ORPO, or GRPO employ different data requirements, and a comparative study across methods would provide additional insights. Third, although we design general curation rules with percentile-based thresholds, we only explore a limited set of quantiles and balancing strategies due to computational constraints. A broader ablation study would help determine optimal thresholds across tasks and model scales. Finally, as UltraMix is derived from existing open datasets, it inevitably inherits their respective coverage gaps and biases.

Broader Impact. By releasing detailed annotations, curated mixtures, and reproducible curation recipes, our work lowers the barrier to systematic research on preference optimization and promotes transparency in corresponding dataset design. Our annotations can be directly reused by researchers and practitioners to probe dataset quality, conduct controlled ablations, or build tailored preference mixtures for specialized domains such as math, code, or reasoning. *UltraMix*, our curated dataset, achieves stronger benchmark performance with 30% fewer samples than TuluDPO, offering both alignment improvements and compute efficiency gains in DPO training. While we apply our recipe to five prominent DPO corpora, the general approach, consisting of quality filtering, reward verification, and task balancing, can be applied to other datasets in principle. Nevertheless, we acknowledge that preference datasets, like any general-purpose alignment corpus, may carry dual-use risks: preference optimization can amplify biases, encode harmful behaviors, or be misapplied in sensitive domains. We therefore encourage responsible use and support future work on adversarial safety evaluations, fairness-aware preference curation, and multilingual extensions of DPO datasets.

Contributions. This work presents the first systematic, sample-level comparison of widely used open-source DPO datasets. We evaluate two representative models across 14 benchmarks covering instruction following, coding, math, and reasoning. Grounded in extensive Magpie-based annotations and preference reward validation, we provide a detailed dissection of TuluDPO, UltraFeedback, ORPO, HelpSteer, and Code-Preference-Pairs, revealing differences in composition, reward reliability, and task specialization. In particular, we show that pre-annotated reward scores do not reflect the assessment of specialized reward models, indicating bias and misalignment in existing DPO datasets. Leveraging these insights, we curate *UltraMix*, a reward-aligned high-quality DPO dataset that consistently outperforms the best-performing TuluDPO corpus while reducing compute requirements. Our methodology, which combines automated annotations, reward-based filtering, and task-aware balancing, offers a reusable framework for future dataset curation. By making both the annotations and UltraMix publicly available, we aim to establish a reproducible foundation for data-centric preference optimization research and accelerate progress in transparent and efficient LLM post-training.