Beyond Sample-Level Feedback: Using Reference-Level Feedback to Guide Data Synthesis

Anonymous ACL submission

Abstract

LLMs demonstrate remarkable capabilities in following natural language instructions, largely due to instruction-tuning on high-quality datasets. While synthetic data generation has emerged as a scalable approach for creating such datasets, maintaining consistent quality standards remains challenging. Recent approaches incorporate feedback to improve data quality, but typically operate at the sample level, generating and applying feedback for each response individually. In this work, we propose REFERENCE-LEVEL FEEDBACK, a novel methodology that instead collects feedback based on high-quality reference samples from carefully curated seed data. We use this feedback to capture rich signals of desirable characteristics and propagate it throughout the data synthesis process. We present REFED, a dataset of 10K instruction-response pairs synthesized using such feedback. We demonstrate the effectiveness of our approach by showing that Llama-3.1-8B-Instruct finetuned on REFED achieves state-of-the-art performance among similar-sized SFT-based models on AlpacaEval 2.0 and strong results on Arena-Hard. Through extensive experiments, we show that our approach consistently outperforms traditional sample-level feedback methods with significantly fewer feedback collections and improves performance across different model architectures.

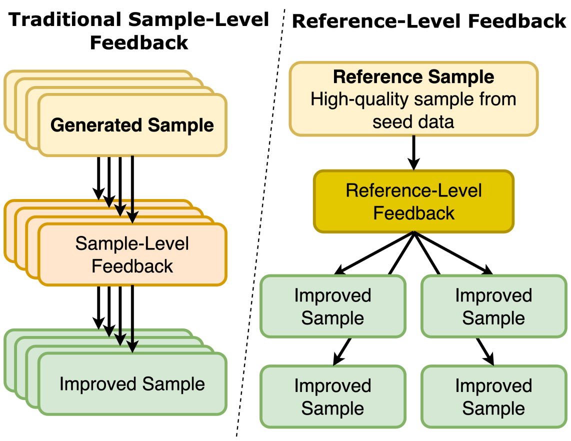

Figure 1: Comparison of feedback approaches for data synthesis. Left: Traditional sample-level feedback generates and applies feedback individually for each sample. Right: Our REFERENCE-LEVEL FEEDBACK approach collects feedback once from a high-quality reference sample and applies it to synthesize and improve multiple new samples.

1 Introduction

Large Language Models (LLMs) demonstrate remarkable capabilities in following natural language instructions and performing real-world tasks (OpenAI et al., 2024b; Dubey et al., 2024). This can be largely attributed to instruction-tuning, which refers to supervised finetuning (SFT) on instruction-response pairs (Wei et al., 2022; Bai et al., 2022a; Ouyang et al., 2022). Recent advancements in instruction-tuning emphasize the importance of high-quality datasets in enhancing model performance (Chen et al., 2024b; Zhou et al., 2023).

Traditionally, high-quality instruction-tuning datasets are created by repurposing existing datasets or using human annotators (Wei et al., 2022; Wang et al., 2022; Ouyang et al., 2022; Zhou et al., 2023; Chen et al., 2024c). However, these methods present challenges that prevent the creation of large-scale datasets, such as data scarcity and the considerable cost and time required for human annotation (Liu et al., 2024; Long et al., 2024; Singh et al., 2024). The use of synthetic data for the creation of instruction-tuning datasets has emerged as a reliable alternative that overcomes such challenges (Wang et al., 2023b; Taori et al., 2023; Xu et al., 2023; Peng et al., 2023).

To further improve the quality of the synthesized data, recent approaches incorporate natural language feedback (Chen et al., 2024d,a; Sun et al., 2024; Bai et al., 2022b). In these approaches, an LLM generates a response to an existing instruction, then collects feedback on their response either through self-reflection, from a stronger LLM or a human annotator. This resulting feedback is provided to the LLM to refine its initial response. Such uses of feedback has proven effective in improving LLM performance on alignment benchmarks as well as reinforcing specific principles such as helpfulness and truthfulness (Chen et al., 2024a; Sun et al., 2024; Bai et al., 2022b). The current feedback-driven approaches operate at the sample-level, which means that feedback is generated for and applied to each response individually.

Rather than collecting sample-level feedback, we propose REFERENCE-LEVEL FEEDBACK, a novel approach that uses feedback collected from the high-quality reference samples in a seed dataset, as shown in Figure 1. Many data synthesis approaches carefully curate reference samples to use as seed data, which are used as in-context examples for synthesis (Wang et al., 2023b; Taori et al., 2023). We extend this and collect feedback based on these reference samples, since they serve as exemplars for training data. These samples are of higher quality than model generated samples, so the captured feedback provides a richer signal towards the desirable characteristics (i.e. clarity, relevance) for a data sample.

Our framework is presented in Figure 2. For each reference sample, we identify the desirable characteristics of both the instruction and response components and use it to create instruction and response feedback, respectively. The instruction-specific feedback is used to guide the synthesis of new instructions, and response-specific feedback is used to refine the corresponding responses. Since synthesized instructions share key characteristics of their reference counterparts, response-specific feedback remains relevant and is used to improve the quality of synthesized responses. This framework enables us to systematically propagate the desirable qualities of reference samples to newly generated samples, establishing overall higher quality standards for data synthesis.

We demonstrate the effectiveness of our approach through **RE**ference-Level **F**eedback **E**nhanced **D**ata (REFED), a dataset synthesized using our framework. Models fine-tuned on REFED achieve state-of-the-art performance on instruction-following benchmarks AlpacaEval 2.0 (Dubois et al., 2024) and Arena-Hard (Li et al., 2024c). Through comprehensive experiments and analyses, we demonstrate that using REFERENCE-LEVEL FEEDBACK is more effective for synthetic data generation compared to existing approaches.

The main contributions of this work are [1]:

- We introduce REFERENCE-LEVEL FEEDBACK for data synthesis, a novel method that leverages feedback collected from reference samples to capture and propagate desirable characteristics to newly synthesized data. Using our framework, and the LIMA (Zhou et al., 2023) training dataset as seed data, we synthesize REFED, a dataset of 10K instruction-response pairs.

- We demonstrate the effectiveness of our approach by presenting Llama-3.1-8B-Instruct-REFED, a Llama-3.1-8B-Instruct model fine-tuned on REFED. The resulting model achieves state-of-the-art performance among similarly sized models trained only with SFT on the AlpacaEval 2.0 leaderboard (Dubois et al., 2024), with a 21.06% improvement in length-controlled win rate. It also shows strong performance on Arena-Hard, outperforming larger models like GPT-3.5-Turbo (Li et al., 2024c).

- Through comprehensive experiments, we demonstrate that our approach: (1) outperforms models trained on other synthetic instruction-tuning datasets, (2) consistently improves base and instruct variants of different model architectures, and (3) provides more effective quality improvements compared to traditional sample-level feedback approaches, while also being more efficient.

2 Related Work

Synthetic Data for Instruction Tuning. Data synthesis has emerged as an effective and scalable approach to creating instruction-tuning datasets. One line of approaches use instruction-response pairs from a seed dataset as in-context examples to guide synthesis (Wang et al., 2023b; Taori et al., 2023; Peng et al., 2023). While Wang et al. (2023b) use models to self-generate their training data, subsequent works leverage more capable proprietary

[1] Our code and data are available at https://anonymous.4open.science/r/anon_refed-DD20

models to generate higher quality data (Taori et al., 2023; Peng et al., 2023).

Other approaches have explored alternative synthesis strategies. There are works that use structured guidance through manually curated taxonomies or LLM-generated skill sets (Li et al., 2024a; Kaur et al., 2024). Xu et al. (2024) uses pre-query templates to sample instructions from aligned LLMs and generate instructions that reflect the LLM's existing knowledge.

Many works have explored methods for enhancing the quality of synthesized data. Xu et al. (2023) proposes Evol-Instruct, which generates increasingly complex versions of existing instructions. Other approaches include using multi-agent simulation (Tang et al., 2024) or incorporating natural language feedback (Bai et al., 2022b; Chen et al., 2024a; Sun et al., 2023).

Natural Language Feedback. Natural language serves as a rich medium for providing feedback to LLMs, effective in conveying detailed and nuanced information. Recent work has demonstrated the effectiveness of using LLMs to generate this feedback. Madaan et al. (2023) introduce Self-Refine, which has LLMs generate feedback and refine their own responses. Following this, several works have shown that using various feedback methods and fine-tuned critic models can yield further improvements (Jin et al., 2023; Wang et al., 2023a; Gou et al., 2024; Wu et al., 2024).

Another application of feedback is at the dataset level, focusing on creating higher-quality training data. Constitutional AI (Bai et al., 2022b) generates self-critiques and revisions to create training data aligned with specific principles. In a similar manner, Self-Align (Sun et al., 2023) uses natural language descriptions of desirable qualities to guide LLMs towards producing more aligned outputs, IterAlign (Chen et al., 2024d) uses an iterative process to discover constitutions and self-correct, and Chen et al. (2024a) demonstrate the effectiveness of feedback-based refinement in code generation tasks.

In order to more effectively incorporate feedback for data synthesis, we introduce REFERENCE-LEVEL FEEDBACK. It fundamentally differs from existing feedback-based methods in three key aspects. First, while previous work collects feedback at the sample-level, we collect feedback from high-quality reference samples in the seed data. This enables us to identify and propagate desirable qualities from reference samples and establish higher quality standards for the data synthesis.

Secondly, our approach more effectively leverages seed datasets. Rather than just using seed data samples as in-context examples for synthesizing similar samples, we systematically analyze and capture the specific qualities that make the reference sample effective.

Lastly, we expand the role of feedback beyond response refinement and guide the entire data synthesis process: our method uses feedback to synthesize new instructions and to refine the corresponding responses.

3 Method

In this section, we present our data synthesis pipeline that leverages REFERENCE-LEVEL FEEDBACK to generate high-quality instruction-response pairs. An overview of the pipeline is presented in Figure 2, and the steps are detailed in the following subsections. Complete examples for each step can be found in Appendix A, and the prompts used for each section can be found in Appendix B.

3.1 Feedback Collection

Our pipeline begins with a seed dataset – a small collection of carefully curated instruction-response pairs that serve as exemplars for synthesized data samples. It can be either manually crafted by human annotators or automatically selected using quality-based criteria. These reference samples are high-quality and exhibit desirable characteristics such as clarity and relevance, which we aim to replicate in our synthetic data. For REFERENCE-LEVEL FEEDBACK, we systematically identify and capture such qualities through a framework that identifies the strength of each sample, as well as potential areas for improvement.

Unlike traditional approaches that collect feedback on generated responses at the sample-level, our method identifies the qualities that make reference samples high-quality and uses it for feedback. This feedback captures a richer signal than feedback collected at the sample-level, establishing higher quality standards for synthesis and providing more effective guidance for generating training data that exhibits similar properties to the reference samples.

For each reference sample in the seed dataset, we collect REFERENCE-LEVEL FEEDBACK from both the instruction and the response:

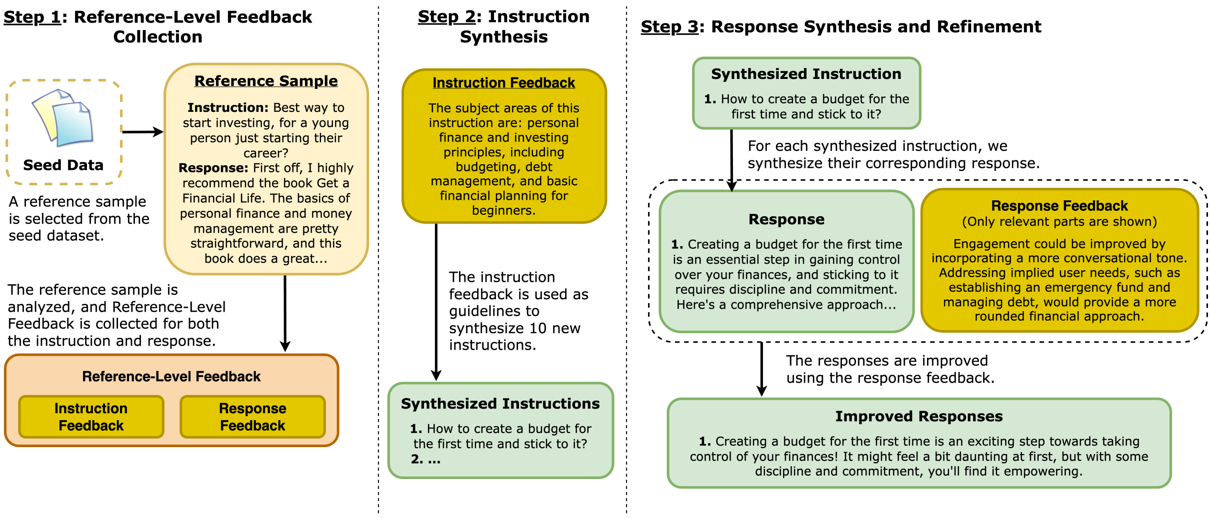

Figure 2: An overview of our data synthesis pipeline. Starting from our seed data, we select a reference sample and collect REFERENCE-LEVEL FEEDBACK on both the instruction and response. Instruction feedback is used to synthesize new instructions. We generate their corresponding responses, and then improve it using the response feedback.

Instruction Feedback. To collect feedback from a reference instruction and capture essential features that make it effective for training, we analyze key attributes (e.g., clarity and actionability). We also ensure comprehensive coverage along a wide breadth by collecting feedback along two dimensions: relevant subject areas (e.g. cellular biology, csv file manipulation, legislative processes) and relevant skills necessary to respond to the instruction (e.g. understanding of specific tools, knowledge of processes, analysis). This enables us to systematically identify desirable characteristics of instructions while maximizing the breadth of instruction types.

Response Feedback. When collecting feedback from a reference response, we identify key qualities that make it an effective response to the instruction. We evaluate along multiple critical dimensions, including factual accuracy, relevance to the instruction, and comprehensiveness. This feedback captures both the strengths of the reference response and specific areas where it can be improved upon.

3.2 Data Synthesis

Now, we use the collected REFERENCE-LEVEL FEEDBACK from the previous stage to synthesize new data samples, while maintaining the quality standards established by our reference data. For each reference sample and its corresponding feedback, we employ a two-phase synthesis process, as illustrated in Figure 2:

1. **Instruction Synthesis.** We provide an LLM the reference instruction as an example and the instruction feedback as guidelines to synthesize new instructions that maintain the qualities specified in the feedback. As depicted in Step 2 of Figure 2, we synthesize 10 new instructions for **subject-based** feedback, which produces instructions that align with the subject areas of the reference response. We also synthesize 10 new instructions for **skill-based** feedback, which produces instructions that align with the skills needed to respond to the reference instruction.

2. **Response Synthesis and Refinement.** For each synthesized instruction, we first generate an initial response. We then enhance this response using the reference response feedback, instructing the language model to analyze the feedback and incorporate the relevant aspects. This process is shown in Step 3 of Figure 2.

Note on relevance of response feedback. Although the response feedback was originally collected for the reference response, many aspects of it can still remain applicable because of the shared characteristics between the reference and synthesized instructions. We acknowledge that not all feedback elements may transfer, and to account for this, we explicitly instruct the model to selectively apply only the relevant aspects of the feedback and ignore the irrelevant aspects. An example of this can be found in A.

This synthesis process enables us to synthesize new data, while systematically propagating the high-quality characteristics of reference samples.

3.3 Theoretical Efficiency Analysis

Our presented pipeline for data synthesis with REFERENCE-LEVEL FEEDBACK is significantly more efficient than using traditional sample-level feedback methods, specifically in the frequency of feedback collection. While sample-level approaches require feedback for every synthesized sample, our method only requires feedback once for every reference sample. This translates to a reduction from $O(n)$ feedback collections, where n represents the number of synthesized samples, to $O(1)$. However, it is also important to note that this efficiency gain comes with an initial fixed cost of collecting and curating seed data.

4 Experiments

4.1 Experimental Setup

Data Synthesis. We use the LIMA (Zhou et al., 2023) training dataset as our seed dataset, which comprises of one thousand carefully curated instruction-response pairs. The samples were either manually written or selected from community forums, and were selected based on quality as well as diversity. This dataset was chosen because it is concise enough to serve as a seed dataset, while being well-designed and has demonstrated effectiveness for instruction tuning (Zhou et al., 2023).

In our experiments, we use GPT-4o mini (OpenAI, 2024) with our data synthesis framework to create REFED, an instruction tuning dataset with 10K data samples.

Training Setup. We finetune the base and instruct variants of Llama-3.1-8B (Grattafiori et al., 2024) and Mistral-7B (Jiang et al., 2023) on REFED. We use a learning rate of 1×10^{-6} for instruct variants, and 2×10^{-5} for base variants. All other hyperparameters remain consistent across models: linear warmup ratio of 0.03, cosine decay, batch size of 128, and maximum sequence length of 2048. The models are trained for 15 epochs, with checkpoint selection based on length-controlled win-rate (Dubois et al., 2024) on a held-out validation set of 100 synthesized instruction-response pairs that were synthesized with GPT-4o (OpenAI et al., 2024a).

When training on larger datasets like Evol Instruct (Xu et al., 2023) and UltraChat (Ding et al., 2023), we follow prior works and modify our training setup as follows: 100 warmup steps, batch size of 32, and train for 2 epochs (Xu et al., 2024).

Evaluation. To evaluate our model's instruction-following abilities, we use two benchmarks: AlpacaEval 2.0 (Dubois et al., 2024) and Arena-Hard (Li et al., 2024c). These benchmarks are automatic evaluators of language models' instruction-following abilities and have demonstrated the highest correlations with human preferences from Chatbot Arena (Li et al., 2024b; Dubois et al., 2024).

Both benchmarks compute win rates by using a powerful LLM as a judge to compare model responses against established responses from a reference model. To further improve correlation with human preferences, AlpacaEval 2.0 additionally computes a length-controlled win rate that mitigates biases towards longer responses by comparing responses of similar length (Dubois et al., 2024).

In our experiments, we follow standard evaluation protocols and use GPT-4-Turbo (1106) as a judge. For AlpacaEval 2.0, we use GPT-4-Turbo (1106) as the reference model, and GPT-4-Turbo (0314) for the reference model in Arena-Hard.

4.2 Experimental Results

4.2.1 Experiment 1: How Effective is REFERENCE-LEVEL FEEDBACK for Data Synthesis?

The first set of experiments evaluate the effectiveness of REFERENCE-LEVEL FEEDBACK for data synthesis by comparing it against traditional sample-level feedback, and also systematically analyzing different components in our framework. For each approach, we synthesize datasets with 10K samples, finetune Llama-3.1-8B-Instruct on that data, then evaluate.

We conduct an ablation study by progressively introducing the different components of the reference-level feedback in our framework, instruction and response feedback. Starting with a baseline of no feedback, we finetune on just our initial seed dataset. Next, we train on a dataset with 10K samples that was created by incorporating instruction feedback from REFERENCE-LEVEL FEEDBACK and generate the corresponding response. Lastly, we evaluate our complete approach by syn-

Feedback Type	AlpacaEval 2				Arena-Hard	
	LC (%)	WR (%)	SE	Len.	WR (%)	Tok.
No Feedback	32.45$_{\uparrow 9.55}$	32.98$_{\uparrow 9.54}$	1.65	2106	29.2$_{\uparrow 7.9}$	873
REF-LEVEL Instruction Feedback	38.99$_{\uparrow 16.09}$	35.34$_{\uparrow 11.90}$	1.68	1926	29.8$_{\uparrow 8.5}$	634
REF-LEVEL Instruction + Sample-Level Response Feedback	42.92$_{\uparrow 20.02}$	41.74$_{\uparrow 18.30}$	1.73	1959	30.8$_{\uparrow 9.5}$	642
REF-LEVEL Instruction + Response Feedback	**43.96**$_{\uparrow 21.06}$	**42.24**$_{\uparrow 18.80}$	**1.74**	**1950**	**35.9**$_{\uparrow 14.6}$	**670**

Table 1: Analysis of the different components of REFERENCE-LEVEL FEEDBACK for data synthesis. We evaluate the impact of the instruction and response feedback, and also compare against traditional sample-level feedback for response improvement, while using reference samples from LIMA (Zhou et al., 2023). Results show performance after finetuning Llama-3.1-8B-Instruct on each generated dataset. Green subscripts indicate improvements after fine-tuning. Metrics shown are: Length-Controlled Win Rate (LC), Win Rate (WR), Standard Error (SE), Average Length (Len.), and Average # Tokens (Tok.).

Model	# Samples	AlpacaEval 2				Arena-Hard	
		LC (%)	WR (%)	SE	Len.	WR (%)	Tok.
Llama-3.1-8B-Instruct	-	22.90	23.44	1.49	2181	21.3	861
+ Alpaca	52K	10.80	4.60	0.72	530	6.6	321
+ Evol Instruct	143K	13.65	6.77	0.88	949	7.0	532
+ UltraChat	208K	13.57	6.52	0.86	853	7.8	500
+ Instruct-SkillMix-SDA	4K	43.31	38.43	1.71	1658	25.2	466
Infinity-Instruct-7M-Gen-Llama3.1-8B	9M	31.62	25.78	1.54	1588	33.1	716
Llama-3-8B-Instruct-SkillMix	4K	38.63	42.98	1.75	4058	12.8	1790
Gpt-3.5-turbo-0613	-	22.35	14.10	1.04	1331	24.8	401
Llama-3.1-405B-Instruct	-	39.26	39.11	1.43	1988	69.3	658
Claude 3 Opus (02/29)	-	40.51	29.11	1.39	1388	60.4	541
Llama-3.1-8B-Instruct-REFED	**10K**	**43.96**	**42.24**	**1.74**	**1950**	**35.9**	**670**

Table 2: Evaluation results of Llama-3.1-8B-Instruct finetuned on REFED against selected baselines (detailed in Section 4.2.2). *Top* shows results from finetuning on various synthetic datasets. *Middle* shows the performance of leading models from AlpacaEval 2.0 leaderboard. *Bottom* shows our model trained on REFED. Results demonstrate that our model outperforms these baselines across both evaluation benchmarks.

thesizing a dataset that also incorporates the response feedback to improve the generated response (REFED).

Additionally, we compare against sample-level feedback, where feedback is generated and applied individually for each response. Here, the synthesis pipeline remains consistent, with minimal prompt modifications to accommodate different feedback types. With this, we can effectively isolate the impact of different feedback strategies on response quality.

Results. The results in Table 1 demonstrate improvements in performance as each component of our framework is introduced. On both benchmarks, we see a clear improvement as we introduce using instruction feedback, and response feedback to synthesize data.

On AlpacaEval 2.0, using the complete REFERENCE-LEVEL FEEDBACK for data synthesis achieves a length-controlled win rate of 43.96% and win rate of 42.25%, showing that it is superior to sample-level feedback (LC: 42.92%, WR: 41.74%). Results on Arena-Hard are similar, where it achieves a win rate of 35.9%, substantially outperforming sample-level feedback (WR: 30.8%).

The consistent performance gains across both benchmarks demonstrate that REFERENCE-LEVEL FEEDBACK is more effective for improving responses and generating high-quality data compared to alternative feedback types.

4.2.2 Experiment 2: How Does Our Method Compare Against Other Baselines?

We evaluate the performance of our synthetic data by comparing a Llama-3.1-8B-Instruct model finetuned on our dataset against several baselines.

Baselines. For the first set of baselines, we finetune Llama-3.1-8B-Instruct on various well-known synthetic datasets: Alpaca (Taori et al., 2023), Evol Instruct (Xu et al., 2023), UltraChat 200K (Ding et al., 2023), and Instruct-Skillmix-SDA (Kaur et al., 2024). We use an identical training setup

Model	AlpacaEval 2				Arena-Hard	
	LC (%)	WR (%)	SE	Len.	WR (%)	Tok.
Mistral-7B-v0.3	-	-	-	-	-	-
+ REFED	16.97	17.70	1.34	2070	3.6	669
Mistral-7B-Instruct-v0.3	20.61	16.69	1.11	1581	12.6	541
+ REFED	41.10$_{\uparrow 20.49}$	40.55$_{\uparrow 23.86}$	1.73	2069	25.0$_{\uparrow 12.4}$	648
Llama-3.1-8B	-	-	-	-	-	-
+ REFED	29.63	30.10	1.62	2095	12.7	633
Llama-3.1-8B-Instruct	22.90	23.44	1.49	2181	21.3	861
+ REFED	**43.96**$_{\uparrow 21.06}$	**42.24**$_{\uparrow 18.80}$	**1.74**	**1950**	**35.9**$_{\uparrow 14.6}$	**670**

Table 3: Evaluation results of finetuning REFED on the base and instruct variants of Llama-3.1-8B and Mistral-7B models. Green subscripts indicate improvements after finetuning. Note that we do not report base model performance because they are not instruction-tuned.

to the one we use for our models.

We also compare against leading models from the AlpacaEval 2.0 leaderboard that use SFT to train 8B-parameter models: Llama-3-8B-Instruct-Skillmix, which trains Llama-3-8B on the Instruct-Skillmix dataset (Kaur et al., 2024), and Infinity-Instruct-7M-Gen-Llama3.1-8B model, trained on Infinity-Instruct-7M and Infinity-Instruct-Gen (BAAI, 2024). Additionally, we consider some larger and more powerful models such as GPT-3.5, Llama-3.1-405B-Instruct (Dubey et al., 2024) and Claude 3 Opus (Anthropic, 2025).

Results. Our results are presented in Table 2. The Llama-3.1-8B-Instruct model finetuned on REFED achieves state-of-the-art performance among similar sized models trained with SFT, across both evaluation benchmarks. On AlpacaEval 2.0, it achieves a length-controlled win rate of 43.96%. This not only scores higher than our selected baselines, but also outperforms significantly larger models including LLama-3.1-405B-Instruct and Claude 3 Opus. On Arena-Hard, we get a win-rate of 35.9%, outperforming both our baseline models and established models like GPT-3.5 Turbo. These results demonstrate that our data synthesis approach can enable strong model performance on established benchmarks, highlighting the effectiveness of REFERENCE-LEVEL FEEDBACK.

4.2.3 Experiment 3: Does REFED Generalize To Different Model Architectures?

In this section, we evaluate the effectiveness of REFED across different models by finetuning both base and instruct variants of Llama-3.1-8B (Grattafiori et al., 2024) and Mistral-7B (Jiang et al., 2023). This analysis validates the robustness of our approach by demonstrating consistent benefits across different cases.

Results. Our results are presented in Table 3. Training on REFED yields improvements across all model variants. In particular, the instruct models show very strong performance. Llama-3.1-8B-Instruct-REFED achieves the strongest performance, with a length-controlled win rate of 43.96% on AlpacaEval 2.0 and 35.9% on Arena-Hard. Mistral-7B-Instruct-REFED shows impressive results, with 41.0% and 25.0% respectively.

The base models also demonstrate notable improvements. Llama-3.1-8B-REFED achieves a length-controlled win rate of 29.63% on AlpacaEval 2.0 and 12.7% on Arena-Hard, outperforming Llama-3.1-8B-Instruct (20.9%) on AlpacaEval 2. Similarly, Mistral-7B-REFED achieves 16.97% on AlpacaEval 2.0 and 3.6% on Arena-Hard, getting close performance to Mistral-7B-Instruct (20.7%).

These results demonstrate that REFED effectively improves instruction-following capabilities across different models and model variants. The strong performance gains, particularly in base models surpassing their instruct variants, highlight how effective our dataset is in developing LLM instruction-following abilities. This observation matches the model-agnostic design of our method.

4.2.4 Experiment 4: Does Filtering Enhance the Effectiveness?

We explore how different filtering approaches affect model performance by finetuning Llama-3.1-8B-Instruct on various subsets of filtered data. We compare three strategies: random sampling, LLM-judge filtering, and ROUGE-L similarity filtering.

Random Sampling. As our baseline, we randomly sample subsets of size 1K, 2K, 4K, and

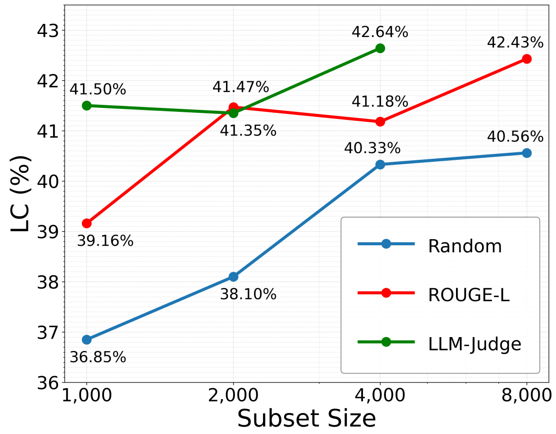

Figure 3: Length Controlled Win-Rate on AlpacaEval 2.0 for Llama-3.1-8B-Instruct finetuned on various subsets of REFED, based on different filtering strategies.

8K from REFED.

LLM-Judge Filtering. We use GPT-4o-mini as a judge to evaluate pairs of initial and refined responses. We only keep samples where refined responses are rated higher than initial responses, and obtain approximately 5K instruction-response pairs. From these, we sample subsets of size 1K, 2K, and 4K.

ROUGE-L Similarity Filtering. Following Wang et al. (2023b), we use ROUGE-L similarity scoring to maximize instruction diversity. Starting with a randomly selected sample, we iteratively add candidates where the instruction's maximum similarity score with existing instructions is below a specific threshold. We use thresholds of 0.10, 0.11, 0.12, and 0.145 to get subsets of sizes 1K, 2K, 4K, and 8K respectively.

Analysis. Figure 3 shows the effects of data filtering. The results demonstrate clear benefits of filtering strategies. LLM-Judge filtering proves most effective, achieving 42.64% performance with just 4K samples – comparable to the full dataset's performance with less than half the data. ROUGE-L similarity filtering performs slightly worse, but still outperforms random sampling, achieving 42.43% with 8K samples. Although neither filtered dataset leads to higher results than the full dataset, they give comparable results while requiring less training time and computational cost. The results suggest that these filtering strategies successfully identify high-quality samples, though the slight drop in performance indicates that filtered-out responses may still contain valuable training signal.

4.3 Empirical Efficiency Analysis

Our method demonstrates significant efficiency advantages in both computational and cost requirements. Using REFERENCE-LEVEL FEEDBACK, we collect feedback from 1K reference samples to synthesize 10K new samples. This means that we collect feedback only 1K times. In contrast, using sample-level feedback would require 11K feedback collections – 1K for instruction synthesis and 10K for response improvement. The reduction in feedback collection, combined with the strong performance metrics, highlights the advantages of our reference-level approach.

Furthermore, we achieve state-of-the-art results without requiring the most expensive language models. While approaches like Kaur et al. (2024) report costs of $600 to synthesize 4K samples using GPT-4, our experiments synthesize 10K samples for less than $20 using GPT-4o-mini. Having such a more cost efficient approach, while also achieving better performance, demonstrates that high-quality data synthesis is possible with more economical models.

5 Conclusion

In this work, we introduce REFERENCE-LEVEL FEEDBACK, a novel framework for enhancing synthetic data quality. Our approach leverages feedback collected from high-quality reference samples to identify and propagate desirable characteristics through the synthesized data. LLMs finetuned on our generated dataset, REFED, achieve strong performance on instruction-following benchmarks. By modeling and incorporating desirable characteristics from carefully curated reference samples, we demonstrate a powerful and efficient approach for creating high-quality instruction-tuning datasets.

Our work opens up several promising directions for future applications. Recent developments in language models like DeepSeek-R1 (DeepSeek-AI et al., 2025) and OpenAI's o1 (OpenAI et al., 2024b) demonstrate that high-quality synthetic data plays a crucial role in achieving state-of-the-art performance. The feedback introduced in our paper can be naturally integrated with various existing data synthesis approaches, such as preference data creation, to further build upon them. By demonstrating the effectiveness of REFERENCE-LEVEL FEEDBACK in this context, this work establishes a foundation for developing more comprehensive approaches for data synthesis.

6 Limitations

Reliance on a Stronger LLM. Our framework relies on a stronger LLM to synthesize data. While this enables the synthesis of high quality data, removing this dependency could help lead to a more robust and independent framework, possibly at the cost of performance degradation. Additionally, LLM-generated data may amplify existing biases or include inappropriate content.

Seed Data Quality. The quality of our synthesized data is tied to that of our seed data. We select concise, high-quality datasets from prior works to use as the seed data. A more comprehensive exploration of seed data selection and its impact on synthetic data remains an important direction for future work.

Furthermore, our work does not fully address the scalability our framework. There likely exists a limit to how much data we can synthesize from our seed data, until the synthesized data becomes repetitive and lacks diversity.

LLM-Based Evaluation. Our evaluation relies on benchmarks that use LLMs as a judge. Although they correlate highly with human judgments, it is important to acknowledge that they may still have limitations, such as biases towards longer responses or their own outputs.

References

Anthropic. 2025. Claude 3 family.

BAAI. 2024. Infinity instruct. *arXiv preprint arXiv:2406.XXXX*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022b. Constitutional ai: Harmlessness from ai feedback. *Preprint*, arXiv:2212.08073.

Angelica Chen, Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. 2024a. Learning from natural language feedback. *Transactions on Machine Learning Research*.

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

Xiusi Chen, Jyun-Yu Jiang, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Wei Wang. 2024c. MinPrompt: Graph-based minimal prompt data augmentation for few-shot question answering. pages 254–266, Bangkok, Thailand.

Xiusi Chen, Hongzhi Wen, Sreyashi Nag, Chen Luo, Qingyu Yin, Ruirui Li, Zheng Li, and Wei Wang. 2024d. Iteralign: Iterative constitutional alignment of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1423–1433.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. pages 3029–3051, Singapore.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. Critic: Large language models can self-correct with tool-interactive critiquing. *Preprint*, arXiv:2305.11738.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary

DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Di Jin, Shikib Mehri, Devamanyu Hazarika, Aishwarya Padmakumar, SUNGJIN LEE, Yang Liu, and Mahdi Namazifar. 2023. Data-efficient alignment of large language models with human feedback through natural language. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Simran Kaur, Simon Park, Anirudh Goyal, and Sanjeev Arora. 2024. Instruct-skillmix: A powerful pipeline for llm instruction tuning. *arXiv preprint arXiv:2408.14774*.

Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, Yuxian Gu, Xin Cheng, Xun Wang, Si-Qing Chen, Li Dong, Wei Lu, Zhifang Sui, Benyou Wang, Wai Lam, and Furu Wei. 2024a. Synthetic data (almost) from scratch: Generalized instruction tuning for language models. *Preprint*, arXiv:2402.13064.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024b. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *Preprint*, arXiv:2406.11939.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024c. From live data to high-quality benchmarks: The arena-hard pipeline.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. 2024. Best practices and lessons learned on synthetic data. *Preprint*, arXiv:2404.07503.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-driven synthetic data generation, curation, and evaluation: A survey. pages 11065–11082, Bangkok, Thailand.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

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. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024b. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2024. Gpt-4o mini: advancing cost-efficient intelligence.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *Preprint*, arXiv:2304.03277.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Preprint*, arXiv:2312.06585.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Advances in Neural Information Processing Systems*, volume 36, pages 2511–2565. Curran Associates, Inc.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2024. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36.

Shuo Tang, Xianghe Pang, Zexi Liu, Bohan Tang, Rui Ye, Xiaowen Dong, Yanfeng Wang, and Siheng Chen. 2024. Synthesizing post-training data for llms through multi-agent simulation. *Preprint*, arXiv:2410.14251.

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

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023a. Shepherd: A critic for language model generation. *Preprint*, arXiv:2308.04592.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. pages 13484–13508, Toronto, Canada.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *Preprint*, arXiv:2406.08464.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021. Curran Associates, Inc.

A Examples

This section presents an example of our data synthesis framework using REFERENCE-LEVEL FEEDBACK. First, we select a reference sample:

Prompt A.1: Reference Sample

Instruction: I have a string representing a unix timestamp (i.e. “1284101485”) in Python, and I’d like to convert it to a readable date. When I use time.strftime, I get a TypeError:

```
»>import time
»>print time.strftime(”%B %d %Y”, ”1284101485”)
Traceback (most recent call last):
File ””, line 1, in
TypeError: argument must be 9-item sequence, not str
```

Response: There are two parts here:

- Convert the unix timestamp (“seconds since epoch”) to the local time
- Display the local time in the desired format.

A portable way to get the local time that works even if the local time zone had a different utc offset in the past and python has no access to the tz database is to use a pytz timezone:

```
#!/usr/bin/env python
from datetime import datetime
import tzlocal # $ pip install tzlocal
unix_timestamp = float(”1284101485”)
local_timezone = tzlocal.get_localzone() # get pytz timezone
local_time = datetime.fromtimestamp(unix_timestamp, local_timezone)
```

To display it, you could use any time format that is supported by your system e.g.:

```
print(local_time.strftime(”%Y-%m-%d %H:%M:%S.%f%z (%Z)”))
print(local_time.strftime(”%B %d %Y”)) # print date in your format
```

If you do not need a local time, to get a readable UTC time instead:

```
utc_time = datetime.utcfromtimestamp (unix_timestamp)
print(utc_time.strftime(”%Y-%m-%d %H:%M:%S.%f+00:00 (UTC)”))
```

If you don’t care about the timezone issues that might affect what date is returned or if python has access to the tz database on your system:

```
local_time = datetime.fromtimestamp (unix_timestamp)
print(local_time.strftime(”%Y-%m-%d %H:%M:%S.%f”))
```

On Python 3, you could get a timezone-aware datetime using only stdlib (the UTC offset may be wrong if python has no access to the tz database on your system e.g., on Windows):

```
#!/usr/bin/env python3
from datetime import datetime, timezone
utc_time = datetime.fromtimestamp (unix_timestamp, timezone.utc)
local_time = utc_time.astimezone()
print(local_time.strftime("%Y-%m-%d %H:%M:%S.%f%z (%Z)"))

Functions from the time module are thin wrappers around the corresponding C
API and therefore they may be less portable than the corresponding datetime
methods otherwise you could use them too:

#!/usr/bin/env python
import time
unix_timestamp = int("1284101485")
utc_time = time.gmtime(unix_timestamp)
local_time = time.localtime(unix_timestamp)
print(time.strftime("%Y-%m-%d %H:%M:%S", local_time))
print(time.strftime("%Y-%m-%d %H:%M:%S+00:00 (UTC)", utc_time))
```

Then, we collect REFERENCE-LEVEL FEEDBACK for that reference sample:

Prompt A.2: Reference-Level Feedback

Instruction Subject Feedback: Programming, specifically Python programming with a focus on date and time manipulation using Unix timestamps.

Instruction Skill Feedback: Understanding of Unix timestamps, Python's datetime and time modules, exception handling in Python, formatting dates and times, knowledge of time zones and portability of code across different systems.

Response Feedback: The reference response is effective in addressing the instruction for multiple reasons. Firstly, it accurately identifies the issue with the TypeError during the use of ‘time.strftime‘ and provides the correct method of converting a unix timestamp into a readable date format using the ‘datetime‘ module, which is more appropriate for this task. The response recognizes the importance of local time and considers timezone issues, which adds depth to the explanation.
Additionally, the structure of the response is clear and organized into distinct sections that guide the user step-by-step through the conversion process. It provides multiple options (using different libraries and methods) for handling the conversion, catering to various user needs, which enhances comprehensiveness.
However, there is room for improvement. 1. **Clarity**: While the response provides various methods, it could improve clarity by explicitly stating under what circumstances each method should be used (e.g., when to use ‘tzlocal‘, when UTC is sufficient, etc.). 2. **Comprehensiveness**: The response could briefly explain what a Unix timestamp is for those unfamiliar with it and its relevance in this context. 3. **Engagement**: Incorporating a more conversational tone or additional commentary about best practices when dealing with date and time conversions could make the response feel more engaging. 4. **Potential errors**: It might be worth noting that ‘pytz‘ needs to be installed and that some users might run into issues if they don't set up their environment beforehand. 5.

Code snippets: Ensure that code snippets are correctly formatted for clarity, especially in online platforms.
Overall, the response effectively meets the instruction but could enhance user understanding and engagement with minor adjustments.

Then, we begin the data synthesis. This process involves generating 10 new instructions for the instruction subject feedback and 10 for the instruction skill feedback. For each instruction, we generate a response and improve the response with the reference response feedback.

Below, we show an example of a new instruction that was generated based on the instruction skill feedback, as well as it's generated response and response improved with the reference response feedback.

Prompt A.3:

Instruction: I am working with a Python script that reads timestamps in milliseconds from a file, but when I try to convert them to a readable format using datetime.utcfromtimestamp, I face a TypeError. My code looks like this:

```
import datetime
timestamps = [1640995200000, 1641081600000]
for ts in timestamps:
print(datetime.utcfromtimestamp(ts))
```

Response: The issue you are encountering is due to the fact that datetime.utcfromtimestamp expects the timestamp to be in seconds, not milliseconds. Since your timestamps are in milliseconds, you need to convert them to seconds by dividing each timestamp by 1000 before passing it to utcfromtimestamp. Here's how you can modify your code to work correctly:

```
import datetime
timestamps = [1640995200000, 1641081600000]
for ts in timestamps:
# Convert milliseconds to seconds
seconds = ts / 1000
# Get the UTC datetime from the seconds
print(datetime. utcfromtimestamp(seconds))
```

In this code:

- We divide each timestamp by 1000 to convert from milliseconds to seconds.
- Then, we call datetime.utcfromtimestamp with the converted value.

When you run this modified code, you should see the output in a readable format without encountering a TypeError:

```
2022-01-01 00:00:00
2022-01-02 00:00:00
```

This output represents the UTC datetime for January 1, 2022, and January 2, 2022, respectively. If you need to format the output differently, you can also use strftime method to customize the output format. For example:

```
for ts in timestamps:
seconds = ts / 1000
utc_time = datetime.utcfromtimestamp(seconds)
formatted_time = utc_time.strftime('%Y-%m-%d %H:%M:%S')
print(formatted_time)
```

Improved Response: The issue you are facing with TypeError is due to the fact that the datetime.utcfromtimestamp function expects timestamps in seconds, not milliseconds. A Unix timestamp represents the number of seconds that have elapsed since January 1, 1970 (known as the Unix Epoch). In your case, the timestamps are in milliseconds, so we need to convert them to seconds by dividing each timestamp by 1000 before passing it to utcfromtimestamp. Here's how you can adjust your code to work correctly:

```
import datetime
timestamps = [1640995200000, 1641081600000]
for ts in timestamps:
# Convert milliseconds to seconds
seconds = ts / 1000
# Get the UTC datetime from the seconds
print(datetime. utcfromtimestamp(seconds))
```

In this modified code:

- Each timestamp is divided by 1000 to convert from milliseconds to seconds.
- Then, we use datetime.utcfromtimestamp to get the UTC time.

When you run this code, you should see the output in a readable format without encountering any TypeError:

```
2022-01-01 00:00:00
2022-01-02 00:00:00
```

This output represents the UTC datetime for January 1, 2022, and January 2, 2022, respectively. If you require the date in a specific format, you can customize it using the strftime method. For instance:

```
for ts in timestamps:
seconds = ts / 1000
utc_time = datetime.utcfromtimestamp(seconds)
formatted_time = utc_time.strftime('%Y-%m-%d %H:%M:%S')
print(formatted_time)
```

Keep in mind that if you're working across different time zones and need to account for local time, you may want to explore libraries like pytz. Just remember that you will need to install it first and ensure your environment is set up correctly. Handling date and time conversions can often be tricky, so it's good practice to decide upfront whether you need UTC or local times based on the context of your application.

After considering the relevant aspects from the reference response feedback, the improved response enhances understanding and engagement. Some changes include: adding an explanation of unix timestamps, explaining when to use UTC and when to use local time, adopting a more conversational tone, and including details of how to setup the environment.

B Prompt Templates

This section presents the prompt templates we use throughout our pipeline.

B.1 Prompts Used For Feedback Collection

Prompt B.1: Reference-Level Instruction Feedback

```
# Task
Analyze the following instruction to extract key features that make it effective for instruction tuning. This analysis will help create high-quality instruction-response pairs for training language models to better follow instructions.

# Context
The analysis will be used to create training data for instruction tuning language models. Focus on qualities that help language models learn to:
- Better understand user intentions
- Recognize instruction patterns
- Generate appropriate responses

# Input
{{ "instruction": {instruction},
"reference_response": {reference_response},
}}
# Output Format
{{ "subject_areas": str, # This should be a description of the relevant subject areas and domains the instruction covers "relevant_skills": str # This should be a description of the relevant skills required to provide a good response to the instruction }}

# Analysis Guidelines
- Consider what makes this instruction clear and actionable
- Identify all relevant domains and skills
- Note structural elements that enhance instruction clarity

Output only a JSON object, in the format specified
```

Prompt B.2: Response Feedback

```
# Task
Analyze the instruction-response pair and provide detailed feedback on how well it addresses the instruction. The feedback should:
- Highlight the specific qualities that make the response effective
- Provide actionable feedback for improvement

# Input
{{ "instruction": {instruction},
"reference_response": {reference_response},
```

```
}}

# Evaluation Criteria
## Content Quality
- Accuracy and factual correctness
- Quality and depth of coverage

## Communication
- Clarity and comprehensiveness
- Logical flow, organization, and structure
- Appropriate quality and depth
- Engagement and tone

## Instruction Alignment
- How will it addresses the instruction
- Appropriate scope and focus
- Match with implied user needs

# Output Format
{{ "response_feedback" : str # Feedback describing strengths of the response and how it can be improved }}

Output only a JSON object, in the format specified.
```

B.2 Prompts Used For Data Synthesis

Prompt B.3: Instruction Synthesis

```
# Task
Generate 10 new instructions based on the provided instruction feature and sample. Each instruction should:
- Be of similar complexity and length to the sample instruction
- Be practical and reasonable to answer
- Be diverse and high-quality

# Sample Instruction:
{instruction}
# Instruction Features:
{feature}

# Output Format
{{ "instructions": list # List of 10 distinct instructions. Each instruction should be a single string. }}

Output only a JSON object, in the format specified.
```

Prompt B.4: Response Synthesis

```
# Task
I will provide an instruction. Generate a high-quality, helpful response to the instruction. The response should demonstrate expertise, clear reasoning,
```

```
and natural language use.

# Response Requirements
- Directly address all aspects of the instruction
- Response should demonstrate clear reasoning and expertise
- Use clear, natural language
- Include examples or evidence when relevant
- Show step-by-step reasoning where appropriate
- Maintain appropriate length and detail level
- Use proper formatting (lists, paragraphs) as needed

Here is an example of a response to an instruction:
# Sample Input Instruction: {sample_instruction}
# Sample Response:
{reference_response}

# Output Format
{{ "response": "The complete response text here" }}
# Input
{{ "instruction": {instruction}, }}

Generate a properly formatted JSON response, as specified by the Output Format,
that addresses this instruction.
```

Prompt B.5: Response Refinement with Reference-Level Feedback

```
# Task
Given an instruction-response pair and feedback, generate an improved version
of the response by applying the feedback. The feedback was given for a similar
but different instruction-response pair. Not all aspects of the feedback may be
directly applicable, so make sure to only apply relevant aspects of the feedback.

# Input
{{ "instruction": {instruction}, "original_response": {response}, "feedback":
{response_feedback} }}

# Quality Assessment Process
1. Analyze Original Response
- Core strengths and effective elements
- Structure and organization
- Depth and comprehensiveness
- Alignment with instruction

2. Evaluate Feedback
- Identify feedback points that are relevant to improving this response, and
ignore points that are not relevant
- Identify actionable improvement suggestions
- Assess potential impact of each change
- Check alignment with original instruction
- Validate that suggested changes maintain or enhance quality
```

```
3. Improvement Strategy
- Prioritize changes with highest impact
- Preserve effective elements of the original response
- Ensure feedback applied enhance the response and do not remove valuable
elements

# Output Format
{{ "analysis": {{
"original_strengths": ["list of key effective elements to preserve"],
"improvement_opportunities": ["list of specific areas that will benefit from
enhancement"], "relevant_feedback": ["list of feedback points that are relevant
and beneficial"] }}, "implementation_strategy": {{
"planned_changes": ["identify what feedback will be applied"], "rationale":
"explain how this feedback will improve the original response" }},
"improved_response": "The revised and improved response" }}

Output only a JSON object, in the format specified.
```

Prompt B.6: Response Refinement with Sample-Level Feedback

```
# Task
Given an instruction-response pair and feedback, generate an improved version
of the response by applying the feedback.

# Input
{{ "instruction": {instruction}, "original_response": {response}, "feedback":
{self_reflection} }}

# Quality Assessment Process
1. Analyze Original Response
- Core strengths and effective elements
- Structure and organization
- Depth and comprehensiveness
- Alignment with instruction

2. Evaluate Feedback
- Identify actionable improvement suggestions
- Assess potential impact of each change
- Check alignment with original instruction
- Validate that suggested changes maintain or enhance quality

3. Improvement Strategy
- Prioritize changes with highest impact
- Preserve effective elements of the original response
- Ensure feedback applied enhance the response and do not remove valuable
elements

# Output Format
{{ "analysis": {{
"original_strengths": ["list of key effective elements to preserve"],
```

```
"improvement_opportunities": ["list of specific areas that will benefit from
enhancement"] }}, "implementation_strategy": {{
"planned_changes": ["identify what feedback will be applied"], "rationale":
"explain how this feedback will improve the original response" }},
"improved_response": "The revised and improved response" }}

Output only a JSON object, in the format specified.
```

C License

Our use of existing artifact(s) is consistent with their intended use. The LIMA dataset follows the CC BY-NC-SA license, or a stricter license if the source data follows the same.