Data Diversity Matters for Robust Instruction Tuning

Alexander Bukharin[1], Shiyang Li[2], Zhengyang Wang[2], Jingfeng Yang[2], Bing Yin[2], Xian Li[2], Chao Zhang[1,2], Tuo Zhao[1,2], Haoming Jiang[2]

[1]Georgia Institute of Technology, [2]Amazon
Correspondence: abukharin3@gatech.edu

Abstract

Recent works have shown that by curating high quality and diverse instruction tuning datasets, we can significantly improve instruction-following capabilities. However, creating such datasets is difficult and most works rely on manual curation or proprietary language models. Automatic data curation is difficult as it is still not clear how we can define diversity for instruction tuning, how diversity and quality depend on one other, and how we can optimize dataset quality and diversity. To resolve these issue, we propose a new algorithm, Quality-Diversity Instruction Tuning (QDIT). QDIT provides a simple method to simultaneously control dataset diversity and quality, allowing us to conduct an in-depth study on the effect of diversity and quality on instruction tuning performance. From this study we draw two key insights (1) there is a natural tradeoff between data diversity and quality and (2) increasing data diversity significantly improves the worst case instruction following performance, therefore improving robustness. We validate the performance of QDIT on several large scale instruction tuning datasets, where we find it can substantially improve worst and average case performance compared to quality-driven data selection.

1 Introduction

Large pre-trained language models have demonstrated a remarkable ability to perform a wide range of natural language processing tasks [Vaswani et al., 2017, Devlin et al., 2018, Radford et al., 2019, He et al., 2020, Brown et al., 2020]. Although these models are powerful, pre-trained models such as GPT-3 can be quite difficult to work with and often do not follow user instructions [Brown et al., 2020]. To unlock instruction-following capabilities, researchers have turned to instruction tuning, in which language models are trained to follow instructions on a small set of example instruction-response pairs [Mishra et al., 2021, Wei et al., 2021, Sanh et al., 2021, Wang et al., 2022a]. Instruction tuning has become extremely popular, as it provides a simple way for researchers to train powerful and aligned language models [Taori et al., 2023, Chiang et al., 2023, Xu et al., 2023].

Although initial works apply instruction tuning to large-scale datasets, it has recently been found

that a small set of well chosen instruction-response pairs is sufficient for good performance. In particular, Zhou et al. [2023] showed that by training on only 1000 instructions manually selected or crafted by experts, superior performance can be achieved compared to training on larger datasets. Training with a small dataset has the added benefits of lowering training costs and enabling faster iteration. Although such manual data selection is not scalable, this work raises an important question: How can we automatically select an instruction tuning dataset?

Recent work on dataset curation have identified two characteristics that an instruction tuning dataset should have: (1) the instruction responses should be high quality [Chen et al., 2023, Peng et al., 2023] and (2) the instructions should cover a wide range of tasks (i.e. be diverse) [Wei et al., 2021, Zhou et al., 2023, Gudibande et al., 2023]. To curate high quality datasets, researchers have used proprietary LLMs to measure the quality of each data point in the dataset and then select only the highest quality data points. To improve dataset diversity, researchers have manually selected instructions that cover a wide range of topics and formats [Zhou et al., 2023, Ivison et al., 2023]. While both of these methods enhance instruction-following capabilities, it is not clear how we can select high quality and diverse datasets without relying on manual curation from human experts, a process that is time-consuming and expensive.

In pursuit of this goal, we propose a new algorithm, QDIT, to measure and optimize the diversity and quality of instruction tuning datasets. QDIT measures diversity using the facility location function [Cornuéjols et al., 1983]. The facility location function provides an intuitive measure of subset diversity, as it essentially measures how well represented each data point in the full dataset is by the data points in the selected subset. With this diversity function and quality functions from prior works, we then define a dataset's quality-diversity score as a simple linear combination of dataset quality and diversity. To optimize the quality-diversity score, QDIT employs a greedy strategy, where the data point that will improve the joint quality-diversity score the most is selected at each time step [Nemhauser et al., 1978]. This procedure is extremely efficient, and can easily scale to datasets with millions of instruction.

QDIT provides an effective way to control the diversity and quality of the instruction tuning dataset, allowing us to conduct an in-depth study of diversity and quality in instruction tuning. From this study we identify two key findings: (1) there is an inherent tradeoff between dataset diversity and dataset quality and (2) improving dataset diversity primarily improves the worst and average case instruction following ability, while not affecting best case instruction following ability much. Based on these finding, we are able to use QDIT to optimize the quality-diversity tradeoff, improving worst case performance while maintaining or improving best case and average performance for robust instruction following. We extensively validate our results on five large-scale instruction tuning datasets.

The rest of this paper is organized as follows: Section 2 covers related literature, Section 3 presents the QDIT algorithm, Section 4 contains our experimental results, and Section 5 concludes our work.

2 Related Work

There have been several works that attempt to improve the quality and diversity of instruction tuning datasets.

◇ **Manual Data Selection.** Several works have shown that superior instruction-following capabilities can be unlocked by carefully selecting and writing instruction-response pairs [Zhou et al., 2023, Touvron et al., 2023, Wang et al., 2023a, Ivison et al., 2023]. To select such data quality and diversity are emphasized, with data from various scientific fields and internet forums being selected. However it is not clear how such datasets can be automatically selected.

◇ **Distilling Closed Models.** To reduce the human effort required for dataset creation, researchers have used powerful proprietary LLMs such as GPT-4 to create instruction tuning datasets [Taori et al., 2023, Peng et al., 2023, Chia et al., 2023]. Again researchers found dataset quality [Peng et al., 2023] and diversity [Xu et al., 2023, Li et al., 2023a] to be most important. Although resulting in powerful datasets, the reliance on proprietary language models in these works is expensive and may raise legal concerns [Wang et al., 2023b].

◇ **Automatic Data Selection for Instruction Tuning.** Due to the aforementioned issues, in this paper we focus on automatic selection of smaller instruction tuning datasets from larger ones. Chen et al. [2023], Dong et al. [2023a] show that by rating the quality of each data point and training on the highest quality data points, downstream performance can be significantly improved. Li et al. [2023b] propose to select instructions based on difficulty. A few works attempt to increase diversity by restricting the distance between selected points to be larger than a given threshold [Wang et al., 2022b, Liu et al., 2023]. We find that this method does not necessarily lead to a significant increase in diversity, and that they are more similar to data de-duplication [Tirumala et al., 2023]. We compare QDIT to similar approaches in our experiments.

Our work is also related to several works that use submodular optimization to increase dataset diversity in natural language processing [Kumar et al., 2019, Kirchhoff and Bilmes, 2014]. To the best of our knowledge, we are the first to use both data quality and diversity in the selection procedure.

3 Method

Before presenting QDIT, we discuss how to quantify instruction diversity and instruction-response quality.

3.1 Quantifying Dataset Diversity and Quality

Given a set $A \subseteq V$, a natural way to measure the diversity of the set A with respect to V is by the facility location function [Cornuéjols et al., 1983]

$$d(A) = \sum_{v \in V} \max_{a \in A} \text{sim}(a, v), \tag{1}$$

where $sim(a, v)$ refers to the similarity of a and v. In QDIT, we use the cosine similarity of instruction embeddings as the similarity function in (1), where the instruction embeddings are computed with sentence transformers [Reimers and Gurevych, 2019]. See Appendix A for more details. Intuitively, we can see that a set A that has a high diversity score $d(A)$ will have an $a \in A$ close to each $v \in V$ and will therefore be representative of the set V.

To measure dataset quality, we follow prior works and measure the quality of each (instruction, response) pair using a large language model such as ChatGPT [Chen et al., 2023] or measure the quality of each data point using a scoring model trained on large amounts of human preference data [Ouyang et al., 2022, Bai et al., 2022]. We refer to such a function with $q(\cdot)$, and measure a dataset's overall quality by averaging the quality score of each data point.

3.2 Quality-Diversity Instruction Tuning

In order to simultaneously control quality and diversity of the selected data, we propose a linear combination of quality and diversity as the quality-diversity (Q-D) score:

$$f(a|A, \alpha) = (1 - \alpha)d(a|A) + \alpha q(a),$$

where $\alpha \in [0, 1]$ is a hyperparameter controlling the tradeoff betwen quality and diversity.

To optimize the Q-D score of the selected data, we consider a greedy algorithm – named QDIT shown in Algorithm 1. Specifically, at each iteration, we select the data point that most increases the Q-D score of the current subset. When $\alpha = 0$, the greedy algorithm achieves the best possible approximation ratio (in the worst case) that a polynomial time algorithm can achieve. See more details in [Nemhauser et al., 1978]. We remark that when $\alpha = 1$, QDIT is reduced to the quality driven selection algorithm proposed in Chen et al. [2023] and when $\alpha = 0$, QDIT is reduced the classical greedy algorithm [Nemhauser et al., 1978].

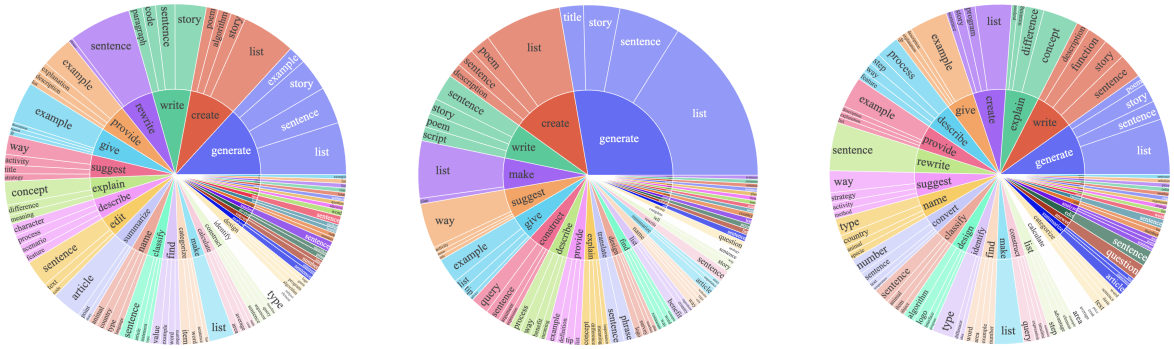
Algorithm 1 QDIT Data Selection: Select a subset of K data points from N data points

Input: K, α

$A \leftarrow \emptyset$
$R \leftarrow V$
$N \leftarrow 0$
while $N < K$ **do**
    $a \leftarrow \text{argmax}_{a \in R} f(a|A, \alpha)$
    $A \leftarrow A \cup \{a\}$
    $R \leftarrow R \setminus a$
    $N \leftarrow N + 1$
end while

Once we finish the selection, we apply instruction tuning on the selected data.

Computational Complexity of QDIT. Finding $a \in A$ that maximizes the quality-diversity score at each time step has complexity $\mathcal{O}(|V|^3)$, leading to a total complexity of $\mathcal{O}(|V|^3 K)$ for QDIT based



(a) Random Selection (b) Quality Driven Selection (c) QDIT ($\alpha = 0$)

Figure 1: Distribution of root verbs and first nouns selected by different algorithms. The dataset size is 3000.

data selection. This is problematic, since for some datasets $|V|$ is greater than 1 million. We take several approaches to reduce data selection cost: (1) parallelization on GPUs, (2) employing the lazy greedy selection algorithm of Minoux [2005], and (3) sub-sampling according to Mirzasoleiman et al. [2015]. These methods can be employed simultaneously, allowing us to select from millions of data points within a few hours. We provide more details in Appendix B, where we find that data selection has minimal cost compared to training.

4 Experiments

We first analyze how the QDIT algorithm affects dataset quality and diversity and then present our main results and analysis.

4.1 Preliminary Analysis

To verify that QDIT can indeed control the quality and diversity of the selected dataset, we first investigate whether the facility location function is aligned with our intuitive understanding of diversity. Next, we study how the α parameter affects the dataset's quality and diversity. For this analysis, we apply the QDIT algorithm to the Alpaca dataset [Taori et al., 2023].

To qualitatively verify that the facility location function is aligned with dataset diversity, we use the Berkeley Neural Parser [Kitaev and Klein, 2018, Kitaev et al., 2019] to extract the root verb and first direct noun from each instruction in the Alpaca dataset. We then plot the distribution of verb-noun pairs in Figure 1 for random, quality driven, and QDIT data selection. From Figure 1, we observe that selecting based on the facility location function indeed improves the dataset diversity compared to random selection, as more verb-noun pairs are included in the dataset and the dataset becomes more uniform. On the other hand, selecting based on quality alone decreases dataset diversity.

Next we plot how α in QDIT affects dataset quality and diversity of 3K selected points in Figure 2. Similar figures for other datasets can be found in Appendix C. From these figures, we can observe

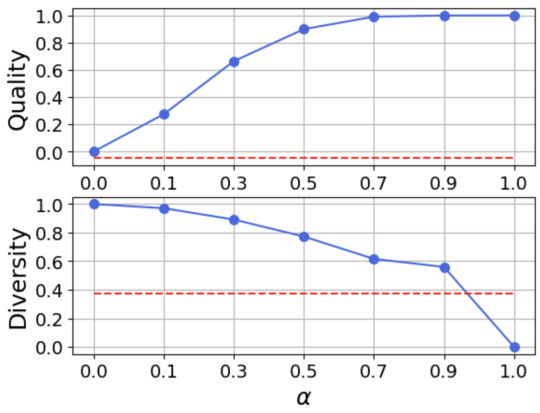


Figure 2: Effect of α on QDIT's dataset quality and diversity. The red line represents a randomly selected dataset.

that there is a tradeoff between quality and diversity, and that QDIT allows us to smoothly control this tradeoff. Moreover, we observe that QDIT is able to improve diversity without significantly decreasing data quality. Altogether, these results indicate that QDIT is a practical way to control dataset diversity and quality.

4.2 Experimental Setup

Now that we can control quality and diversity with QDIT, we seek to study how dataset quality and diversity affect instruction following ability.

◇ **Training Setup.** We use QDIT on two small instruction tuning datasets: Alpaca 52K [Taori et al., 2023], and Dolly 15K [Conover et al., 2023] as well as three large scale datasets: Ultrachat 1.3M [Ding et al., 2023], LMSYS-Chat 1M [Zheng et al., 2023], and a combined dataset of Alpaca 52K, Dolly 15K , and the OIG-small-chip2 dataset (210K). We refer to this dataset as "Mixed 270K". For each large dataset we select a dataset size of 10K points. For each small dataset size we follow the small data setting from Chen et al. [2023] and select $\sim 5\%$ of the original dataset.

To measure instruction-response quality, we use the provided ChatGPT quality scores from Chen et al. [2023] for Alpaca and for all other datasets we use the reward model from Dong et al. [2023b], which is trained on the Anthropic Helpful Harmless dataset and achieves a test accuracy of over 75%. For our main experiments we follow the training procedure from Taori et al. [2023] and use LLaMA-1 7B as our base model [Touvron et al., 2023]. In all settings we use the same number of training epochs, meaning that the training cost is proportional to the number of training instructions. Complete hyperparameter details can be found in Appendix D.

◇ **Evaluation.** We evaluate the trained models in two main ways: through LLM-based pairwise comparison [Dubois et al., 2023] and by using a reward model. For pairwise comparison, we evaluate the trained model versus a variety of reference models, employing Claude 2 as the judge on five evaluation sets: InstructEval [Wang et al., 2022b], WizardLM [Xu et al., 2023], Vicuna [Chiang et al., 2023], Koala [Geng et al., 2023], and a set of 200 examples manually curated from the

ShareGPT dataset. In order to mitigate the effects of the judge LLM's positional bias, we evaluate the responses in both orders (i.e. QDIT response shown first and QDIT response shown second). We then follow Chen et al. [2023] and measure performance according to winning score ($\frac{\#\text{ Win}-\#\text{ Lose}}{\text{Total comparisons}} + 1$). Detailed comparison plots can be found in Appendix E. In addition to language model based evaluation, we evaluate our models based on the reward score achieved on each evaluation dataset. We refer to this score as "HH Score."

◇ **Evaluating Robustness.** Beyond evaluating average performance on the test dataset, we also evaluate the worst and best case performance of each model. We can evaluate the worst case performance with the HH Score by calculating the average score achieved on the worst 10% of instructions for each model (note that the worst instructions can change depending on the model). Similarly, we can evaluate the worst case performance with pairwise comparison by measuring the winning score on the hardest 10% of prompts according to HH score. Best case performance is measured in a similar manner. Measuring best and worst case performance provides more detailed insights into how diversity and quality affect model robustness.

◇ **Baselines.** We primarily compare the QDIT algorithm with two baselines: random data selection and quality based selection. For the Alpaca dataset, the quality baseline is trained on the same data as Alpagasus [Chen et al., 2023].

Table 1: Instruction Tuning results. Random 50K refers to a model trained on a randomly sampled set of 50K points on the corresponding dataset. The top and bottom 10% winning and losing score is computed versus the Alpaca 52K.

Ultrachat 1.3M	**Average Performance**			**Worst Case Performance**		**Best Case Performance**	
	Winning Score vs. Alpaca 52K ↑	**Winning Score vs. Random 50K ↑**	**HH Score Mean ↑**	**Lowest 10% HH Score ↑**	**Lowest 10% Winning Score ↑**	**Top 10% HH Score↑**	**Top 10% Winning Score ↑**
Random 10K	1.138	0.969	6.219	2.620	1.074	9.526	1.167
Quality 10K	1.224	1.013	6.961	3.405	1.175	**10.454**	**1.303**
QDIT 10K	**1.226**	**1.038**	**6.993**	**3.497**	**1.280**	**10.454**	1.293
Mixed 270K	**Average Performance**			**Worst Case Performance**		**Best Case Performance**	
	Winning Score vs. Alpaca 52K ↑	**Winning Score vs. Random 50K ↑**	**HH Score Mean ↑**	**Lowest 10% HH Score ↑**	**Lowest 10% Winning Score ↑**	**Top 10% HH Score↑**	**Top 10% Winning Score ↑**
Random 10K	0.899	0.986	5.443	2.260	0.940	8.80	0.896
Quality 10K	0.959	1.04	6.140	2.973	**0.989**	9.670	0.984
QDIT 10K	**0.987**	**1.083**	**6.276**	**3.054**	0.973	**9.676**	**1.044**
LMSYS 1M	**Average Performance**			**Worst Case Performance**		**Best Case Performance**	
	Winning Score vs. Alpaca 52K ↑	**Winning Score vs. Random 50K ↑**	**HH Score Mean ↑**	**Lowest 10% HH Score ↑**	**Lowest 10% Winning Score ↑**	**Top 10% HH Score↑**	**Top 10% Winning Score ↑**
Random 10K	1.113	1.0	6.176	2.575	1.057	9.589	1.187
Quality 10K	1.198	1.137	**7.066**	3.391	1.052	**10.39**	1.284
QDIT 10K	**1.224**	**1.149**	7.0	**3.551**	**1.149**	10.34	**1.343**

4.3 Main Results

The main experimental results can be found in Table 1 for large datasets and Table 2 for small datasets.

Effect of Quality. Similar to prior works, we find that selecting data based on quality significantly improves average performance, improving the average winning score versus Alpaca 52K by 6.37%

Table 2: Instruction Tuning results on small datasets. The top and bottom 10% winning and losing score is computed versus the Alpaca 52K.

Alpaca 52K	**Average Performance**			**Worst Case Performance**		**Best Case Performance**	
	Winning Score vs. Alpaca 52K ↑	**Winning Score vs. Random 50K ↑**	**HH Score Mean ↑**	**Lowest 10% HH Score ↑**	**Lowest 10% Winning Score ↑**	**Top 10% HH Score↑**	**Top 10% Winning Score ↑**
Random 3K	0.929	-	5.486	2.105	**0.931**	8.986	0.937
Quality 3K	0.920	-	5.629	2.306	0.873	**9.073**	0.934
QDIT 3K	**1.026**	-	**5.661**	**2.513**	0.924	8.791	**1.056**
Dolly 15K	**Average Performance**			**Worst Case Performance**		**Best Case Performance**	
	Winning Score vs. Alpaca 52K ↑	**Winning Score vs. Random 15K ↑**	**HH Score Mean ↑**	**Lowest 10% HH Score ↑**	**Lowest 10% Winning Score ↑**	**Top 10% HH Score↑**	**Top 10% Winning Score ↑**
Random 1K	0.64	0.72	4.632	1.474	0.632	6.144	0.645
Quality 1K	0.71	0.83	5.389	2.082	0.681	7.751	0.746
QDIT 1K	**0.739**	**0.874**	**5.495**	**2.229**	**0.742**	**7.873**	**0.872**

and the average HH score by 11.6% when compared to random selection. However, we also find that selecting based on quality alone can hurt worst case performance, decreasing the lowest 10% winning score in two out of five settings compared to random data selection. We hypothesize that this performance drop is due to the fact that quality based selection hurts dataset diversity.

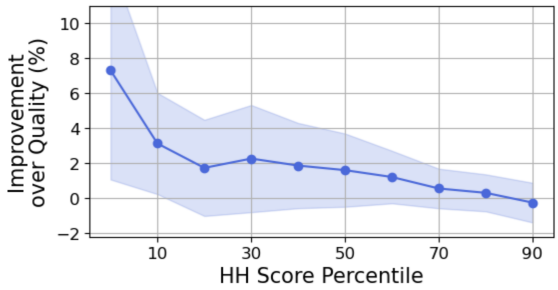
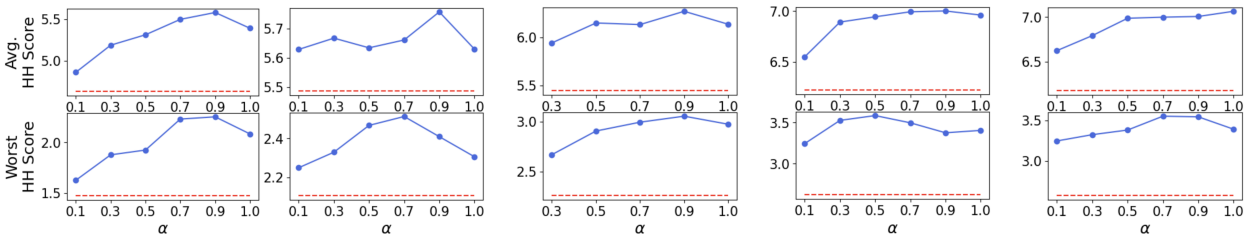


Figure 3: Percent improvement in HH Score of QDIT over Quality-based selection. Performance is averaged over the five datasets and the shaded area represents one standard deviation.

Effect of Diversity. On the other hand, we find that data selection with QDIT is able to achieve both a high average HH score (QDIT improves upon quality driven selection by 4.17% for winning score vs Alpaca 52K and improves average HH score by 1.5%) while achieving a much better worst case performance than both random and quality-driven selection. In particular, we find that QDIT improves worst case HH score by 5.2% and worst winning score vs Alpaca 52K by 6.26% when compared to quality-driven data selection. This trend can be seen in Figure 3, where QDIT improves most over quality selection in the lowest and middle percentiles, while not affecting the highest percentiles. We hypothesize that QDIT's more diverse dataset teaches the model to respond to a wide range of instructions, thereby decreasing the probability that it fails to follow evaluation instructions.

From these experiments we conclude that by increasing data diversity while maintaining data quality, QDIT can improve instruction following capability compared to quality-driven selection. We remark that improvements in worst-case performance are typically more important than improvements in best-case performance, as a high worst case performance will ensure users have a



(a) Dolly 1K (b) Alpaca 3K (c) Mixed 10K (d) Ultrachat 10K (e) LMSYS 10K

Figure 4: Effect of α on best case and worst case performance. The red line represents a randomly selected dataset and $\alpha = 1.0$ is quality-driven data selection. Worst HH Score refers to the bottom 10 percent of HH scores.

consistently positive experience.

Table 3: Evaluation on benchmark datasets. We bold the best result out of quality based selection and QDIT based selection.

	Ultrachat 10K							**LMSYS 10K**						
	MMLU	BBH	ARC	DROP	LAMBADA	SCIQ	AVG	MMLU	BBH	ARC	DROP	LAMBADA	SCIQ	AVG
Random	32.12	33.19	58.34	26.24	69.77	85.4	50.84	33.05	32.57	60.15	25.06	68.5	86.7	51.01
Quality	35.44	32.06	60.34	17.01	**70.38**	85.8	50.17	34.74	32.32	58.54	25.95	68.58	82.6	50.46
QDIT	**36.13**	**32.12**	**60.71**	**26.73**	69.8	**86.8**	**52.05**	**37.34**	**32.52**	**61.44**	**26.41**	**69.28**	**85.0**	**52.0**
	Alpaca 3K							**Mixed 10K**						
	MMLU	BBH	ARC	DROP	LAMBADA	SCIQ	AVG	MMLU	BBH	ARC	DROP	LAMBADA	SCIQ	AVG
Random	36.17	30.25	61.67	26.32	71.64	87.0	52.18	32.93	30.92	58.34	20.33	68.1	84.1	49.12
Quality	34.71	29.97	60.99	19.62	**69.85**	82.7	49.64	33.07	**31.38**	60.34	**26.37**	69.4	88.4	51.49
QDIT	**35.47**	**30.44**	**61.95**	**27.02**	69.68	**84.1**	**51.44**	**34.29**	31.23	**60.71**	26.00	**69.72**	**89.8**	**51.96**

	Dolly 1K						
	MMLU	BBH	ARC	DROP	LAMBADA	SCIQ	AVG
Random	28.11	27.27	59.39	17.26	71.74	80.7	47.41
Quality	33.61	29.95	**60.43**	**24.69**	72.22	**82.8**	**50.62**
QDIT	**33.78**	**30.33**	59.84	22.59	**72.26**	80.6	49.9

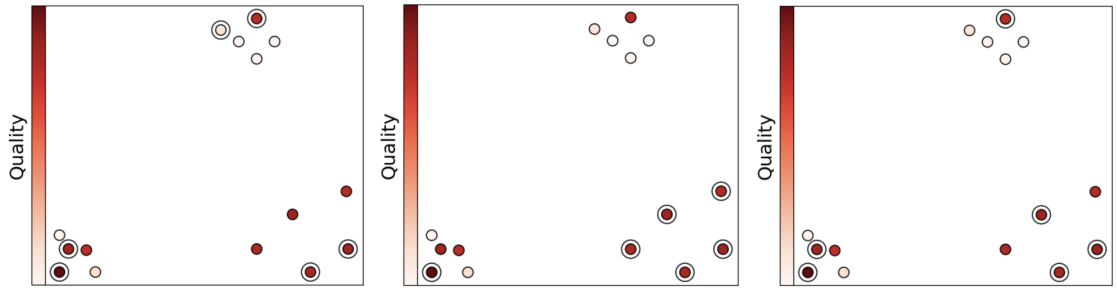
4.4 Analysis

Effect of α. We plot the effect of α on average and worst case performance in Figure 4. From Figure 4, we can see that the performance of QDIT changes smoothly with respect to α, indicating that QDIT is relatively robust to the value of α. In particular, values of $\alpha \in \{0.5, 0.7, 0.9\}$ typically having the highest worst case and average performance. However, decreasing α by too much (i.e. $\alpha = 0.1$) will result in a significant drop in performance, as will increasing α by too much ($\alpha = 1$). This highlights the need for a careful tradeoff between dataset quality and diversity.

Benchmark Performance. For a more comprehensive evaluation of QDIT, we follow Chia et al. [2023] and Gao et al. [2021] by evaluating our model on various benchmark datasets including MMLU [Hendrycks et al., 2020], BBH [Suzgun et al., 2022], DROP [Dua et al., 2019], ARC [Clark et al., 2018], LAMBADA [Paperno et al., 2016], and SCIQ [Welbl et al., 2017]. The results on

these benchmarks can be found in Table 3. Details on our evaluation strategy can be found in Appendix G. Although these benchmarks are not fully aligned with instruction following ability, we find that QDIT typically improves benchmark performance compared to quality-driven selection, achieving a higher average score on four out of five datasets.

Different Base Models. We evaluate the performance of QDIT with different base models in Figure ??. From Figure ?? we find that QDIT can improve both average performance and worst case performance for other base models, including the more powerful LLama-2-13B. We remark that LLama-2-13B is only trained for 2 epochs, possibly resulting in a lower HH score.



(a) QDIT:Cluster (b) QDIT:Threshold (c) QDIT

Figure 5: The selection strategy of the various QDIT algorithms on an example dataset. Six data points are selected and the selected data points are circled.

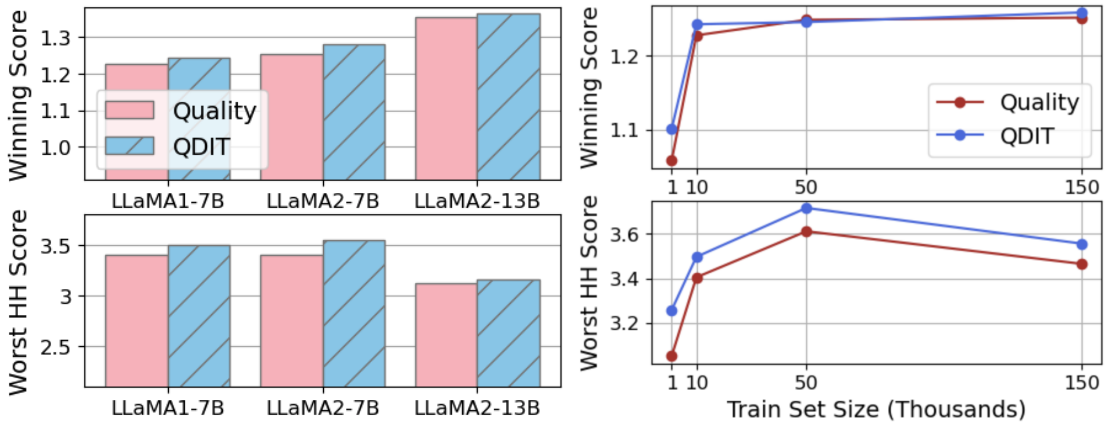


Figure 6: (a) Performance of QDIT with different base models. The dataset is Ultrachat. (b) Ablation with different data sizes. Worst HH Score refers to bottom 10% HH Score. The dataset is Ultrachat.

Data Size. We evaluate the performance of QDIT with different training data sizes in Figure ??, where we find that QDIT leads to the highest gains in low-data regimes, but can still improve performance for larger datasets. We also find that increasing the dataset size beyond $50K$ only results in marginal gains of average performance, and can even decrease worst case HH score. This is likely due to the fact that the average reward score of the dataset decreases as the dataset size increases.

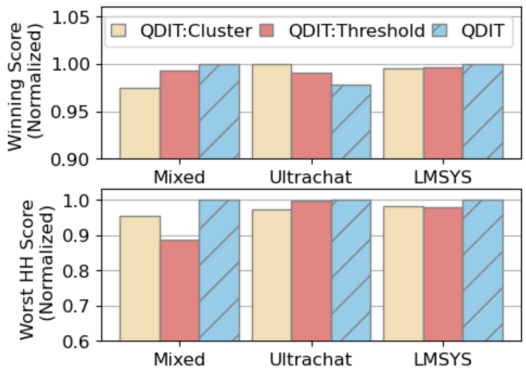
Figure 7: Case study on instruction generalization. This first column shows the test instruction, and the second column shows the most similar instruction in the respective training dataset. The third row shows the model outputs, where quality-based selection simply repeats the video requirements and QDIT-based selection provides useful video suggestions.

QDIT Variants. We study two variants of the QDIT algorithm that also attempt to improve data quality and diversity. The first variant, QDIT:Threshold, first orders the data point by quality, and then iteratively selects instructions that do not have a similarity score greater than τ with any point included selected subset. This algorithm essentially de-duplicates the dataset, and is similar to algorithms used in Wang et al. [2022b] and Liu et al. [2023]. The second variant we propose is QDIT:Cluster, in which we first cluster the instructions and then select an equal amount of data points from each cluster in a quality-driven manner. Comprehensive details can be found in Appendix H.

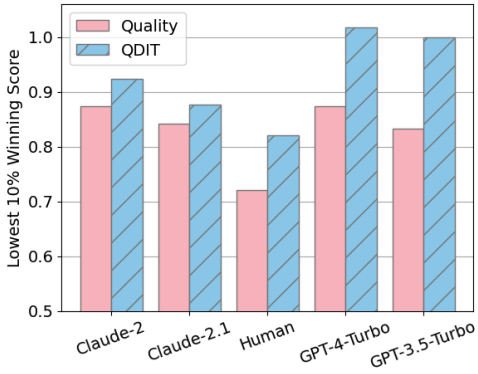
We demonstrate how each variant selects algorithms in Figure 5. From these figures, we can see that the clustering-based approach may not work well as some clusters may be low quality, resulting in low-quality points being chosen. Moreover, the success of this variant is highly dependent on the success of clustering, which is difficult on large and imbalanced instruction tuning datasets. On the other hand, QDIT:Threshold will succeed in de-duplicating the dataset, but may fail to select a sufficiently diverse subset.

Experimentally, we observe in Figure 8a that the two variants of QDIT achieve slightly worse average performance compared to QDIT, but they often result in much worse worst-case performance.

Different Evaluators. To further investigate our finding that data diversity can improve instruction following performance, we compute the worst-case winning score with different evaluators. Concretely, we use Claude-2, Claude-2.1, GPT-3.5-Turbo, and GPT-4-Turbo as different model based evaluators [Achiam et al., 2023]. We also conduct a blind human preference study, using the authors as annotators. Details on the human study can be found in Appendix F. The results using different models as evaluators can be seen in Figure 8b. From these results we can see QDIT consistently improves worst-case performance, further validating our finding that data diversity improves robust instruction following capabilities.



(a) Performance of different QDIT variants on Ultrachat.



(b) Performance with different evaluators. The dataset is Alpaca.

Reducing the Train-Test Gap. One explanation for QDIT's improvement in instruction following capabilities is a reduction in the gap between the training and testing data. More concretely, a model trained on a diverse dataset will be exposed to training instructions similar to those in the test set, and will therefore perform better on the test set.

A qualitative example of this phenomena can be found in Figure 7. In this example, the test instruction asks for help creating a short video. The closest example (measured by cosine similarity of the instruction embedding) in the quality-based dataset is unrelated to this task, while the closest instruction in the QDIT-based dataset is a similar instruction asking how to create a short video on nutrition. As an end result, the QDIT model provides useful video suggestions, while the Quality-driven model merely repeats the provided video requirements.

In all of our experiments we observe that QDIT selects datasets that better cover the testing dataset compared to quality-based selection. This can be seen in Figure 9, where we observe that the QDIT selects more similar instruction to the test instructions than quality-based selection does. This phenomenon provides one explanation on how QDIT improves instruction following ability, but in general it is difficult to directly attribute instruction following capabilities to test-set coverage.

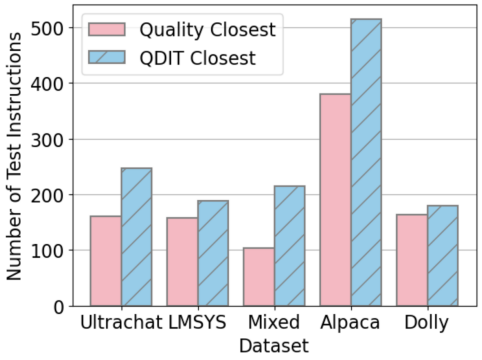


Figure 9: Number of test instructions where quality based selection or QDIT contains the closest training example. Ties are not included.

5 Conclusion

In this paper we introduce QDIT, an automatic method for selecting high quality and diverse instruction tuning datasets. QDIT allows us to flexibly tradeoff quality and diversity, improving both average performance and model robustness on a wide variety of datasets. As a future line of research, we would like to investigate how different similarity metrics affect QDIT's performance.

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

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

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022a.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

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

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.

Gérard Cornuéjols, George Nemhauser, and Laurence Wolsey. The uncapicitated facility location problem. Technical report, Cornell University Operations Research and Industrial Engineering, 1983.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14:265–294, 1978.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*, 2023a.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. Instructeval: Towards holistic evaluation of instruction-tuned large language models. *arXiv preprint arXiv:2306.04757*, 2023.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*, 2023a.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*, 2023b.

Yi Dong, Zhilin Wang, Makesh Narsimhan Sreedhar, Xianchao Wu, and Oleksii Kuchaiev. Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf. *arXiv preprint arXiv:2310.05344*, 2023a.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023b.

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

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S Morcos. D4: Improving llm pretraining via document de-duplication and diversification. *arXiv preprint arXiv:2308.12284*, 2023.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619, 2019.

Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 131–141, 2014.

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

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

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*, pages 234–243. Springer, 2005.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1249. URL https://www.aclweb.org/anthology/P18-1249.

Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1340. URL https://www.aclweb.org/anthology/P19-1340.

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

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023b.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. Blog post, April 2023. URL https://bair.berkeley.edu/blog/2023/04/03/koala/.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 2021.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

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 preprint arXiv:1803.05457*, 2018.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

A Similarity Metrics

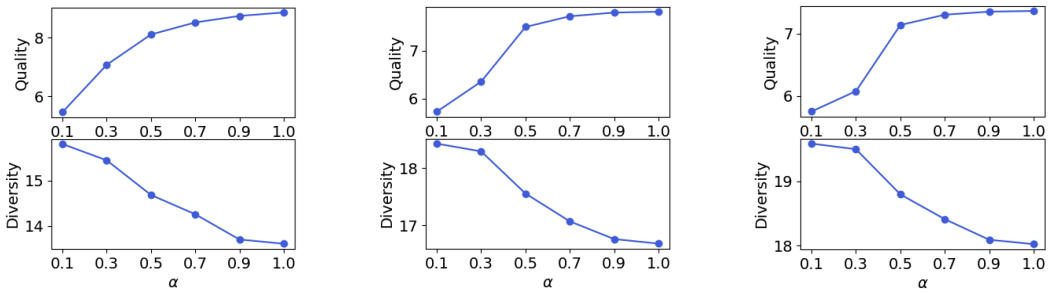
In QDIT, we use the cosine similarity of instruction embeddings as the similarity function in (1), where the instruction embeddings are computed with sentence transformers [Reimers and Gurevych, 2019]. More specifically, we use the all-mpnet-base-v2 model available from Huggingface.

B Computational Cost of QDIT

We measure the wall time of QDIT as taking approximately 9.42 minutes to select a dataset of size 10000. This experiment was conducted on a single A100 40G GPU. In contrast, training on Ultrachat 10K takes 48 minutes on 8 A100 40G GPUs, meaning that data selection (given the embeddings and reward score) has 2% of the cost of training. Taking into account the inference and evaluation process, our method has an even smaller relative cost. We remark that the reward score and embedding generation can typically be done on a small GPU very quickly, or it can even be done on a CPU for minimal cost.

C Tradeoff between Quality and Diversity: Additional Results

We display additional results on the effect of α on different datasets diversity and quality in Figures 10, 11, 12. We again find that there is a tradeoff between quality and diversity, and that α can be used to control this tradeoff.



(a) 1K Selected Points (b) 5K Selected Points (c) 10K Selected Points

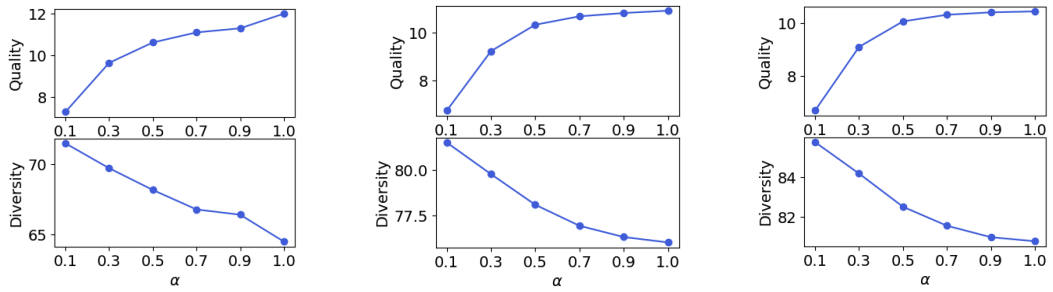
Figure 10: Effect of α on dataset quality and diversity. The dataset is Mixed 270K.

D Training Details

We display the hyperparameter details in Table 4 and Table 5.

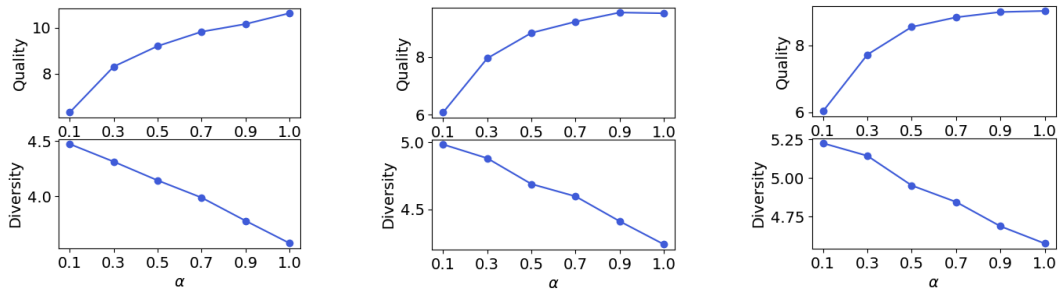
E Complete Win-Tie-Lose Results

In this appendix we show the win, ties, and losses achieved versus the reference models as judged by Claude 2. The results can be seen in Figures 13, 14, 15, 16, 17, and 18.



(a) 1K Selected Points (b) 5K Selected Points (c) 10K Selected Points

Figure 11: Effect of α on dataset quality and diversity. The dataset is Ultrachat 1.3M.



(a) 1K Selected Points (b) 5K Selected Points (c) 10K Selected Points

Figure 12: Effect of α on dataset quality and diversity. The dataset is LMSYS 1M.

Table 4: General training details. The same hyperaparameter setting is used for every dataset and data selection strategy, following Chen et al. [2023].

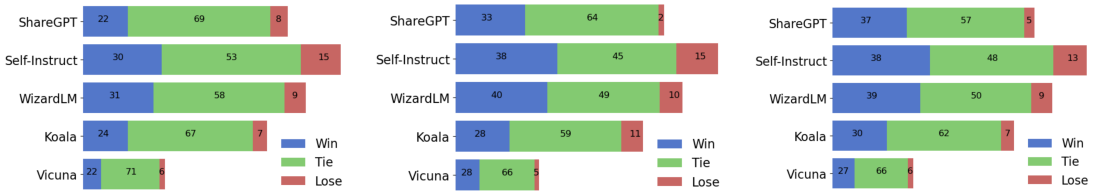
Batch Size	Learning Rate	Epochs	Max Length	Weight Decay
128	2×10^{-5}	3	512	0

Table 5: The hyperparameter α used in the main experiments.

Data Size	Dolly 15K	Alpaca 52K	Mixed 270K	Ultrachat 1.3M	LMSYS 1M
1K	0.7	-	-	-	-
3K	-	0.7	-	-	-
10K	-	-	0.9	-	-
10K	-	-	-	0.7	-
10K	-	-	-	-	0.7

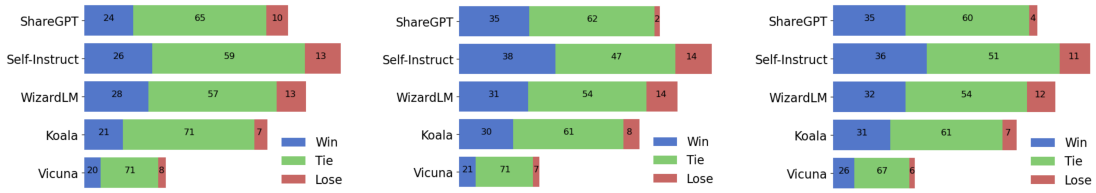
F Human Study Details

For the human study, we use a subset of the paper authors as annotators. For each example, we randomize the order of the reference model generation and evaluated model generation, to keep



(a) Random 10K vs. Alpaca 52K (b) Quality 10K vs. Alpaca 52K (c) QDIT 10K vs. Alpaca 52K

Figure 13: We display the results on Ultrachat as judged by Claude 2. The base model here is LLaMA-2 7B.



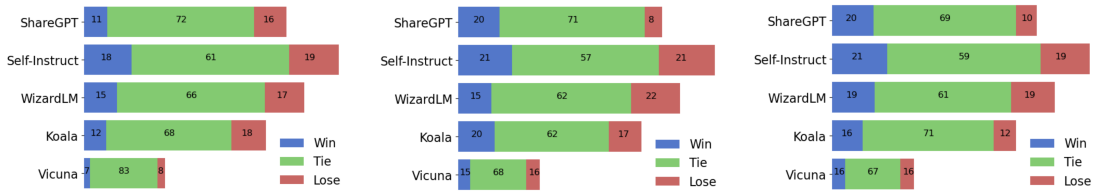
(a) Random 10K vs. Alpaca 52K (b) Quality 10K vs. Alpaca 52K (c) QDIT 10K vs. Alpaca 52K

Figure 14: We display the results on Ultrachat as judged by Claude 2. The base model here is LLaMA-1 7B.

the study blind. We then ask the annotators to select their preferred generation according to the Alpacafarm prompt [Dubois et al., 2023].

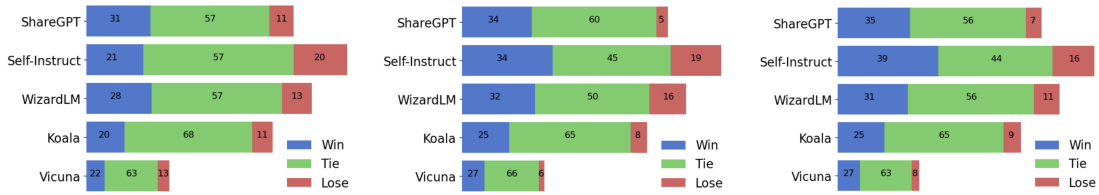
G Benchmark Environments

For evaluation on common NLP benchmarks, we follow Chia et al. [2023]. In particular, we conduct five shot evaluation on MMLU, three shot on BBH, and three shot on DROP. For ARC, LAMBADA, and SCIQ, we use the default zero shot setting of [Gao et al., 2021].



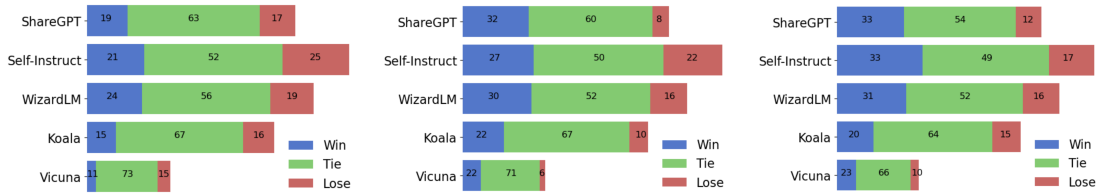
(a) Random 10K vs. Random 50K (b) Quality 10K vs. Random 50K (c) QDIT 10K vs. Random 50K

Figure 15: We display the results on Ultrachat as judged by Claude 2. The base model here is LLaMA-1 7B.



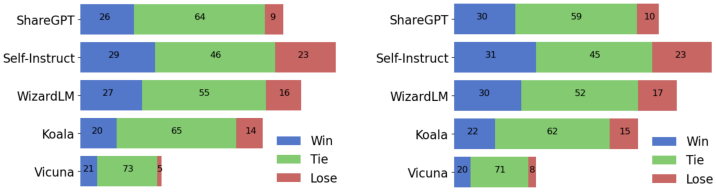
(a) Random 10K vs. Alpaca 52K (b) Quality 10K vs. Alpaca 52K (c) QDIT 10K vs. Alpaca 52K

Figure 16: We display the results on LMSYS as judged by Claude 2. The base model here is LLaMA-1 7B.



(a) Random 10K vs. Random 50K (b) Quality 10K vs. Random 50K (c) QDIT 10K vs. Random 50K

Figure 17: We display the results on LMSYS as judged by Claude 2. The base model here is LLaMA-1 7B.



(a) Quality 10K vs. Alpaca 52K (b) QDIT 10K vs. Alpaca 52K

Figure 18: We display the results on Ultrachat as judged by Claude 2. The base model here is Mistral 7B.

H QDIT Variants

In this section we describe and analyze the variants of QDIT.

QDIT:Cluster. In this variant of QDIT, we first cluster all the instructions based on their sentence-transformer embeddings using the k-means algorithm, where $k = 100$. We then select an equal number of points from each cluster, and points are selected from each cluster based on quality.

QDIT:Threshold. In this variant of QDIT, we first sort the instructions based on quality. We then iteratively remove instructions from the selected dataset if they have a similarity with some other instruction in the dataset greater than a threshold τ. In our experiments we use the cosine similarity as the similarity metric and set $\tau = 0.5$.