ResoFilter: Fine-grained Synthetic Data Filtering for Large Language Models through Data-Parameter Resonance Analysis

Anonymous ACL submission

Abstract

Large language models (LLMs) have shown remarkable effectiveness across various domains, with data augmentation methods utilizing GPT for synthetic data generation becoming prevalent. However, the quality and utility of augmented data remain questionable, and current methods lack clear metrics for evaluating data characteristics. To address these challenges, we propose **ResoFilter**, a novel method that integrates models, data, and tasks to refine datasets. ResoFilter leverages the fine-tuning process to obtain Data-Parameter features for data selection, offering improved interpretability by representing data characteristics through model weights. Our experiments demonstrate that ResoFilter achieves comparable results to full-scale fine-tuning using only **half the data** in mathematical tasks and exhibits strong generalization across **different models and domains**. This method provides valuable insights for constructing synthetic datasets and evaluating high-quality data, offering a promising solution for enhancing data augmentation techniques and improving training dataset quality for LLMs. For reproducibility, we will release our code and data upon acceptance.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across various domains, with training data playing a pivotal role in enhancing their performance. The quality and quantity of data are crucial factors in all stages of LLM development, including pretraining, instruction tuning, and alignment (Peters et al., 2018; Radford and Narasimhan, 2018; Devlin et al., 2019; Raffel et al., 2020; Touvron et al., 2023; Mishra et al., 2021; Sanh et al., 2022; Longpre et al., 2023; Muennighoff et al., 2024; Ziegler et al., 2019; Bai et al., 2022; Ouyang et al., 2022; Rafailov et al., 2023).

The significance of high-quality datasets has been well-established in the pretraining phase, where data cleaning techniques have shown to substantially improve model performance, especially for smaller models (Conneau and Lample, 2019; Raffel et al., 2020; Wenzek et al., 2020; Gao et al., 2020; Rae et al., 2021; Lee et al., 2022). Following the success of ChatGPT (Ouyang et al., 2022; OpenAI et al., 2024), the focus has shifted towards creating high-quality fine-tuning datasets, leading to a surge in methods for automatic generation of instruction-following data using GPT models.

While these methods, such as Self-Instruct (Wang et al., 2023b), Evol-Instruct (Xu et al., 2023), and others (Honovich et al., 2023; Dai et al., 2023; Abdullin et al., 2024), have significantly increased the scale of available datasets, the notion of "high quality" remains contentious. As observed by Schimanski et al. (2024), current approaches predominantly focus on augmenting data volume rather than enhancing quality. This trend has led to the phenomenon of "diminishing returns," where performance gains plateau as dataset size increases beyond a certain threshold, emphasizing the need to prioritize data quality over quantity.

To address this challenge, we propose ResoFilter, a novel method that leverages the fine-tuning process for effective data selection. ResoFilter processes each data point through full forward and backward propagation, capturing the induced change in model weights. From these changes, we derive a characteristic score for each data point, which serves as the metric for subsequent selection. Our approach offers improved interpretability by representing data characteristics through model weights, building upon existing research demonstrating that model weights store knowledge (Hanna et al., 2023; Dai et al., 2022). Figure 1 illustrates the detailed workflow of our method.

Empirical results demonstrate the efficacy of ResoFilter. In the context of MetaMath (Yu et al., 2023), we achieve comparable performance to full

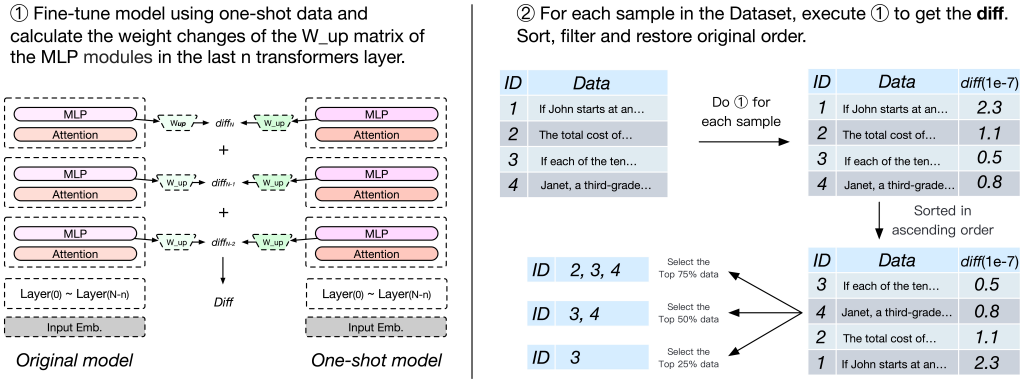

Figure 1: Workflow for our method. The left side of the figure illustrates the detailed process of calculating parameter changes for individual data samples using the W_{up} matrix from the last n layers of the neural network. The right side demonstrates the application of this method to the entire dataset, including steps for computing parameter change values for each sample, sorting based on these values, filtering out samples with the largest parameter changes, and restoring the remaining data to its original order.

dataset fine-tuning using only 50% of the data selected by our method. Moreover, by eliminating poorly performing data points, ResoFilter can even surpass the performance of full fine-tuning. The method also exhibits strong generalization across different models and domains, including mathematics, code, and general question answering tasks.

Our contributions can be summarized as follows:

1. We introduce ResoFilter, an effective method for selecting high-quality datasets from large-scale data collections to enhance the fine-tuning process of large language models.

2. We validate the generalizability of ResoFilter by demonstrating its excellent performance across various domains (including mathematics, code, and general tasks) and different model architectures, showcasing its robustness and wide applicability.

3. Through extensive experimental analysis, we provide valuable insights into the construction of synthetic datasets and the analysis of high-quality data, offering guidance for future data synthesis and selection methodologies.

2 Related Work

Instruction data selection method Using high-quality data for instruction tuning of large language models can significantly enhance their performance. The existing work on selecting high-quality data can be divided into two main parts: (1).Utilizing human expertise or more powerful generative models to assist in the selection process. For example, Lima (Zhou et al., 2024) manually curated a dataset of 1,000 instruction samples for instruction tuning and achieved outstanding performance on the evaluation set. AlpaGasus (Chen et al., 2023b), InsTag (Lu et al., 2023), and DEITA (Liu et al., 2023) leverage the superior performance of ChatGPT to replace manual effort in filtering and selecting data. (2).Model-guided data selection involves filtering data by observing changes in the model or by introducing additional smaller models. (Kung et al., 2023) proposes a new framework based on instantaneous uncertainty, which identifies informative tasks by measuring changes in the probability distribution of the model's outputs. Superfiltering (Li et al., 2024b) utilizes smaller language models to filter out better data. By using GPT-2 combined with Perplexity and IFD (Li et al., 2023a) to filter the data.

Data attribution The work of data attribution aims to quantify the contribution of each training sample to the model's prediction results. It attempts to answer the question, "Which training samples have the most significant impact on a specific prediction result?" In recent studies, (Li et al., 2023a) quantifies the degree of assistance provided by instructions to models by comparing the difficulty of generating responses with and without the instruction context. Nuggets (Li et al., 2023b) leverage the downstream task benefits brought by data to the model for data filtering. LESS (Xia et al., 2024) estimates the impact of data by using gradients in the optimizer and performs low-rank gradient similarity searches for instruction data selection.

Knowledge in LLM From the perspective of model interpretability, it has been consistently observed that language models based on the transformer architecture encode "lower-level" information in the earlier layers and more "semantic" information in the later layers (Tenney, 2019). Researches on the content knowledge of models (Meng et al., 2022; Li et al., 2024a) have emphasized the role of intermediate and top layers in factual prediction, as well as specific heads in ensuring truthfulness. DoLA (Chuang et al., 2023) utilizes the fact-oriented characteristics of the top layers in LLMs to propose contrastive decoding, aiming to mitigate the hallucination problem during the generation process.

All these methods have their certain flaws. **Instruction data selection** methods still introduce biases from the auxiliary models used in the selection process though they have reduce the need for human intervention. **Data attribution** often require complex computations or additional fine-tuning steps. Building on these insights, our work addresses these limitations by proposing a novel approach that directly analyzes the weight differences between the fine-tuned and original models. By focusing on the weight differences in these layers, we can more effectively identify data that contributes to meaningful semantic changes in the model, rather than surface-level or lower-order modifications. This approach allows us to select data that is more likely to improve the model's overall performance and generalization capabilities.

3 Method

3.1 Background and Rationale

Recent advancements in data selection methods focus on improving data representation and effectively combining these representations for data selection (Liu et al., 2024). While some approaches leverage GPT-4 scoring directly (Cao et al., 2023; Wei et al., 2023), we argue for methods that integrate diverse model features for enhanced interpretability (Xia et al., 2024).

Research on model interpretability has revealed the importance of MLP layers and deep layers in feature learning (Hanna et al., 2023; Dai et al., 2022). Additionally, studies on model editing (Mishra et al., 2024) and distillation (Yao et al., 2024) have explored knowledge distribution across various network layers. Building on these insights, we propose a framework that integrates the model, data, and task.

3.2 Problem Definition and Modeling

In the context of Supervised Fine-Tuning (SFT), we aim to balance data richness and its impact on the model for specific tasks. We formulate this as an optimization problem with the following objective function:

$$E(D, M, p) = F_{richness}(p) \cdot (1 + \beta \cdot F_{characteristic}(p)) \quad (1)$$

where D is the dataset, M is the model, and p is the filtering percentage.

Objective Function Design The multiplicative form ensures interdependence between data richness and characteristic intensity. The term $(1 + \beta \cdot F_{characteristic})$ provides a baseline guarantee, ensuring that data richness is always considered even when characteristic intensity is low. This design captures the non-linear relationship between data quantity and quality, which is more realistic than a simple linear combination.

As the filtering percentage p increases, $F_{richness}(p)$ grows, representing the benefit of including more data, while $F_{characteristic}(p)$ captures the relevance of the selected data subset to the specific model and task.

The parameter β modulates the influence of characteristic intensity relative to data richness, while λ in $F_{richness}(p) = 1 - e^{-\lambda|D_p|}$ controls how quickly data richness approaches its maximum value. These parameters enable our problem formulation to adapt to diverse datasets and fine-tuning scenarios within the SFT framework.

Importantly, this objective function captures a crucial trade-off: as p increases, E reaches an optimal point, balancing data quantity and quality. Beyond this point, although performance may decline, increased richness mitigates the impact of decreasing characteristic intensity, preventing catastrophic drops in effectiveness across larger data subset sizes.

Data Richness Modeling We model data richness using an exponential function:

$$F_{richness}(p) = 1 - e^{-\lambda|D_p|} \quad (2)$$

This function is bounded between 0 and 1, providing a normalized measure of richness. It increases monotonically as more data points are included, while exhibiting diminishing returns. This design reflects the real-world scenario where the marginal benefit of additional data decreases as the dataset grows, approaching a theoretical maximum richness as $|D_p|$ tends to infinity.

Characteristic Intensity The characteristic intensity measures the contribution of the selected data subset to the model on a given task:

$$F_{characteristic}(p) = \frac{1}{|D_p|} \sum_{j \in D_p} f(j, M) \quad (3)$$

Here, $f(j, M)$ quantifies the impact of data point j on the model M. This function captures the unique characteristics of each data point in relation to the model.

3.3 ResoFilter

Our proposed method, ResoFilter, comprises two key components: a data screening process and a characteristic intensity calculation. The detailed algorithm is presented in Appendix D.

Data Screening Process We select a subset of data, D_p, based on a ranking function:

$$D_p = \{j \in D \mid \text{rank}(s(j, M)) > |D| \cdot (1 - p/100)\} \quad (4)$$

Here, $s(j, M)$ is a scoring function that ranks each data point j based on its relevance to the model M. We consider data points causing smaller differences in the last n layers of the model as potentially more valuable, as they are less likely to disrupt previously acquired knowledge.

Characteristic Intensity Calculation The function $f(j, M)$ evaluates the impact of each data point on the model's last n layers, ensuring selected data is both relevant and representative of key dataset features.

4 Experiments

4.1 Experiments Setup

Training datasets We conducted experiments separately on datasets in the following three different domains: general domain, code domain, and mathematics domain. In the general domain, two datasets DOLLY (Conover et al., 2023) and OPEN ASSISTANT (Köpf et al., 2024) are included. In the domains of coding and mathematics, we use dataset evol-codealpaca-v1 (Luo et al., 2023) and dataset MetaMath (Yu et al., 2023) respectively. The detailed data structure format can be referred to in Appendix A.

Evaluation We follow the evaluation methodology of Open Instruct (Wang et al., 2023a), where MMLU (Hendrycks et al., 2020) and BBH (Suzgun et al., 2022) are used to assess the model's performance in general domains. We used GSM8k (Cobbe et al., 2021) to evaluate the model's performance in mathematical reasoning. For the code evaluation, we employed the HumanEval dataset (Chen et al., 2021) to assess the model's ability to generate functionally correct programs from docstrings.In accordance with the settings of Open Instruct (Wang et al., 2023a), we refer to this dataset as HumanEval-CodeX. For all experimental tasks, we used the greedy decoding method to obtain the generated results from the model.

Baselines We compare ResoFilter with several baseline methods. In the experiments, we applied each data filtering method to produce three different amounts of training data for model training, specifically 25%, 50%, and 75% of the total data. The simplest method is random, which randomly samples different amounts of training data from the dataset. For the Loss and PPL methods, we first ranked the data from high to low based on loss and PPL, and then selected the top 25%, 50%, and 75% for experiments. Similarly, for Superfiltering (Li et al., 2024b) and Nuggets (Li et al., 2023b), we filtered the data according to their respective methods, resulting in different amounts of training data. We used the last 3 layers diff data and took mean value of them as the filter score.

Models and Training Parameters We conducted experiments using four models: Llama2-7B-Base (Touvron et al., 2023), Gemma2-2B-Base (Team et al., 2024), Mistral-7B-Base-v0.3 (Team, 2024), and Phi2-3.8B-Base- (Javaheripi et al., 2023). In the general domain, experiments were carried out with model Llama2-7B-Base. For the code domain, model Gemma2-2B-Base was utilized. In the mathematics domain, all four models were tested. We standardized the training hyperparameters, with the learning rate set

Model	Method	25%	50%	75%	Full SFT	Time Cost(Hour)
gemma2-2B	Random	0.6042	0.6254	0.6474	0.649	-
	Loss	0.6034	0.6436	0.6436		0.8
	PPL	0.6035	0.6467	0.655		0.8
	Superfiltering	0.5079	0.6262	0.6482		0.8
	Nuggest	0.5565	0.6164	0.6543		**98**
	Our method	**0.6042**	**0.6497**	**0.6603**		1.5
Llama2-7B	Random	0.5525	0.5921	0.5981	0.5731	-
	Loss	0.5034	0.5413	0.5459		1.7
	PPL	0.511	0.5353	0.5451		1.7
	Superfiltering	0.3972	0.6073	**0.6531**		2
	Our method	**0.5549**	**0.6196**	0.6444		4.5
Llama2-13B	Random	0.6276	0.6747	0.6686	0.6935	-
	Loss	0.6096	0.6209	0.6641		4.9
	PPL	0.6141	0.6338	0.6611		4.9
	Superfiltering	0.5496	0.677	0.6831		5.5
	Our method	**0.6322**	**0.6853**	**0.6855**		9

Table 1: Figure 1 presents a comprehensive comparison of our proposed method against several baseline approaches across three different model architectures: gemma-2B, Llama2-7B, and Llama2-13B. In addition to the three most commonly used types of methods, we also cited two methods for data selection (Li et al., 2024c) (Li et al., 2024b). As can be seen, our performance has achieved the best among these methods with only a small increase in time cost compared to other techniques.

	MaxLegnth	BatchSize	Epoch
General Data	2048	16	3
Math Data	768	64	1
Code Data	2048	16	1

Table 2: Detail parameter settings for model training.

to 1e-5 and using the AdamW optimizer. Other training parameters are shown in Table 2.

4.2 Main Results

Method Comparision To evaluate the effectiveness and transferability of our method across different model architectures, we conducted experiments on three models: gemma-2B, Llama2-7B, and Llama2-13B. As shown in Table 1, our method consistently outperforms other selection approaches across all models. Notably, our method gets the best performance in 25% among all the methods in all models, and for gemma-2B and Llama2-7B, our method achieves superior performance to full SFT at both 50% and 75%, demonstrating its ability to maintain high performance with reduced data. Moreover, our approach incurs minimal additional time cost compared to other methods, making it both effective and efficient. These results highlight the robustness and adaptability of our weight difference-based data selection strategy across various model sizes and architectures.

Fields	Method	25%	50%	75%	Full SFT
CodeX	Random	0.2979	0.3549	**0.404**	0.4098
	Loss	0.3146	0.3442	0.3872	
	Our method	**0.3579**	**0.425**	0.3966	
MMLU	Random	0.5105	**0.5152**	0.4858	0.4658
	Loss	0.507	0.501	**0.4961**	
	Our method	**0.5129**	0.4893	0.4952	
BBH	Random	0.3796	**0.3759**	**0.3824**	0.3704
	Loss	0.3796	0.3667	0.3685	
	Our method	**0.3861**	0.362	0.3778	

Table 3: Cross-domain evaluation results comparing our proposed method with random sampling. There has been significant improvement in code generation while in general domain it is not obvious. We have more detail experiments and explanations in Appendix B

Generalizability To assess the generalizability of our proposed method, we conducted experiments across various domains, including general knowledge (MMLU), code generation (HumanEval-CodeX), and reasoning tasks (BBH). Table 3 presents the results of these cross-domain evaluations.

Our method consistently outperforms random sampling across different domains and model architectures, demonstrating its transferability. Notably, in the code domain (HumanEval-CodeX), our approach achieves significant improvements over random sampling, particularly at lower percentages of data selection (p25 and p50).

Methods	25%	50%	75%
Random	0.6042	0.6254	0.6474
Mean	0.6042	0.6497	0.6603
Std	0.5812	0.6406	0.6459
p90	0.5754	**0.6504**	**0.6686**
p95	0.5928	0.6285	0.6542
p99	**0.6156**	0.6474	0.6611
cosine	0.5914	0.6179	0.6361
pearson	0.5898	0.6391	0.6346

Table 4: Here are the GSM8K scores applied various statistical indicators to the W_{up} weight layer of the model, including 5 main statistical methods shown in the table. The p99 achieves the best performance in 25% while p90 get the best in 50% and 75%.

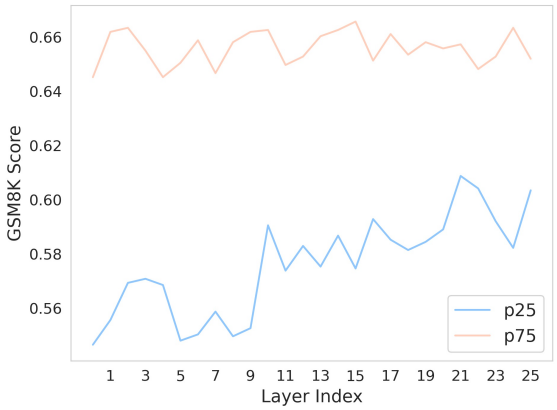

Figure 2: We analyzed the W_{up} weights of the model from the first layer to the 26th layer. The 25% shows a continuous upward trend, while the 75% fluctuates within a certain range.

It's worth noting that for MMLU, both our method and random sampling show a decrease in performance compared to the base model. This is likely due to the base model's strong initial capabilities in general knowledge tasks, combined with the potential mismatch or lower quality of the fine-tuning data. As detailed in Appendix A, this phenomenon highlights the importance of data quality in fine-tuning, especially for models with high baseline performance in specific domains.

5 Discussion

5.1 Alabtion

To thoroughly investigate the robustness and effectiveness of our proposed method, we conduct a series of ablation studies. These experiments aim to provide a comprehensive understanding of various factors influencing the performance of our approach. Unless otherwise specified, we use the mean difference of weights in the last three layers as our default method for the following experiments. We explore the effects of different modules, layer indexes, statistical methods, training data number, and data ordering on our method's performance. The following subsections detail each of these aspects, offering insights into the key components and sensitivities of our approach.

Statistical Methods The results in Table 4 show that different statistical methods perform differently under different data ratios. At 25%, the p99 percentile method performs the best; At 50% and 75%, the p90 percentile method achieved the best results, significantly better than the random baseline. The mean method performs stably at all data ratios and consistently outperforms random selection. In contrast, the performance of cosine similarity and Pearson correlation coefficient methods is relatively weak. These findings indicate that selecting appropriate statistical methods is crucial for effective data screening, especially percentile and mean methods, which demonstrate strong robustness and effectiveness in this task.

Layer Position As shown in Figure 2 showed that on 25% of the data, the model performance showed a significant trend with the change of layers. Shallow (1-8 layers) filtered data resulted in lower GSM8k scores, while deep (20-26 layers) filtered data produced higher scores. This indicates that data with significant differences in deep weights may better capture features that affect task performance. It is worth noting that the 25% curve shows a clear upward trend, while the 75% curve is relatively stable, indicating that for 75%, there is not much difference between deep screening and shallow screening. This difference implies changes in data sensitivity at different levels, providing some reference for optimizing data selection strategies.

Train Dataset Number The results in Figure 3 show that as the number of train dataset increases, the scores of both methods (Diff and Random) show an upward trend. Prior to P25, Random is higher than our method, demonstrating that diversity is more important than quality when the data volume is small. Subsequently, the two curves intersected, and the performance of the Diff method(Ours) was generally better than that of the

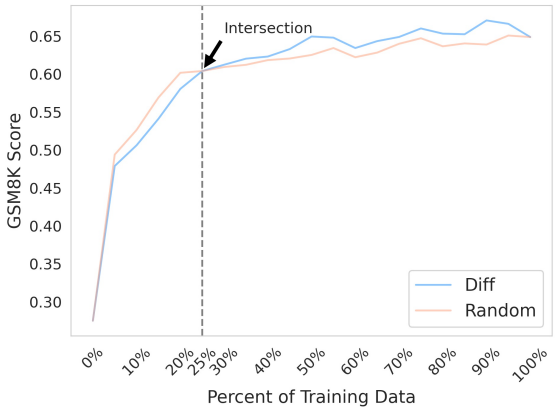

Figure 3: Here we used filtering based on different proportions, and then trained and evaluated using the remaining data. The figure shows the GSM8K scores under different filtering ratios.

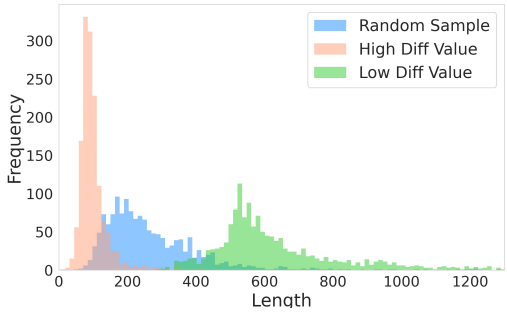

Figure 4: Illustrates the token length distribution comparison across three datasets. Low Diff Value has much larger token numbers than High Diff Value data.

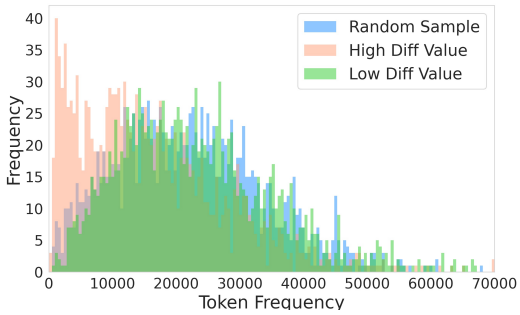

Figure 5: Illustrates the token Frequency distribution comparison across three datasets. In order to analyze the frequency distribution of tokens, we first perform token statistics on the entire dataset to obtain the word frequency of each token. Subsequently, we calculated the total frequency of all token words in each sample divided by the sample length to obtain the average token frequency. High Diff Value shows more obscure vocabulary is used compared to Low Diff Value.

Random method. This indicates that when there is basic diversity, using weight differences to select higher quality data can help achieve better results. As the amount of training data increases, the gap between the two lines also widens, eventually reaching its maximum value (around 3 points) at p90 and p95.

5.2 Feature Analysis

To better understand the characteristics of data samples that lead to different fine-tuning outcomes, we conducted a comprehensive analysis of the High Diff Value, Low Diff Value, and Random Sample datasets. Our analysis focused on the Gemma-2b model, utilizing the mean difference of the last three weight layers as the selection criterion. The High Diff Value set i.e "good sample" is the top 1% of samples with the highest mean difference, while the Low Diff Value set i.e "dirty sample" means the bottom 1% with the lowest mean difference. For comparison, we also included a Random Sample set. We examned these datasets across four key dimensions as shown in the following paragraphs. This analysis aims to uncover the underlying patterns that distinguish high-performing from low-performing fine-tuning data.

Token Lengths Distribution As shown in the Figure 4, the token length distribution of the High Diff Value dataset is significantly shorter where concentrated on the left side of the chart, indicating that these samples usually contain fewer tokens. In contrast, the distribution of the Low Diff Value dataset is more dispersed and tends towards longer token sequences, with its peak appearing in the middle right part. Random samples exhibit a relatively uniform distribution, mainly concentrated in the medium length range.

Token Frequency As shown in the Figure 5, the High Diff Value data exhibits significant peaks in the low-frequency region, indicating that this type of sample tends to use rare or special vocabulary. In contrast, the distribution of Low Diff Value data is more uniform, covering a wider frequency range, suggesting that these samples use more common vocabulary. Random Sample falls between the two and presents a relatively balanced distribution.

Unique Token Ratio As shown in the Figure 6, the three types of data exhibit significantly different distribution characteristics. The unique token ratio

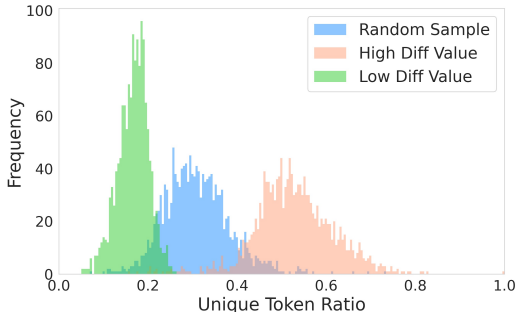

Figure 6: To evaluate the lexical diversity of the dataset, we calculated the ratio of unique tokens in each sample, which is the number of unique tokens in each sample divided by the total number of tokens in the sample. Similar to Figure 4, High Diff Value data shows every sample contains more unique words.

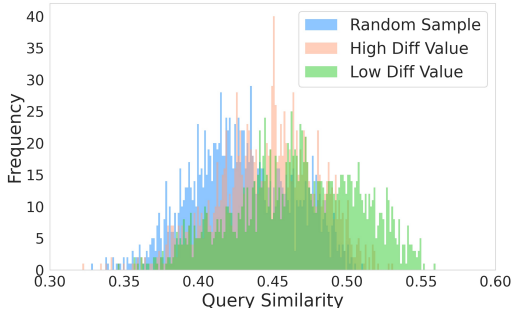

Figure 7: In order to gain a deeper understanding of the characteristics of High Diff Value and Low Diff Value data, we analyzed the similarity distribution between texts. We use a pretrained embedding model (Chen et al., 2023a) to calculate the cosine similarity between each query and the rest queries in the same class, and take the average. This obtained the average similarity distribution of three types of data. Low Diff Value data are much similar each other than else.

of Low Diff Value samples is concentrated in a lower range (0.1-0.3), indicating that these samples tend to use more repetitive vocabulary. In contrast, the distribution of High Diff Value samples tends towards higher ratios (0.4-0.7), indicating that these samples contain more unique vocabulary. Random Sample falls between the two, with a relatively uniform distribution.

Cosine Similarity Distribution As shown in the Figure 7, the similarity distribution of the three types of data shows significant differences. The distribution of Low Diff Value data tends to the right, indicating a high degree of query similarity within this type of data, concentrated in the range of 0.45-0.55. In contrast, the distribution of High Diff Value data is skewed towards the left, mainly in the range of 0.35-0.45, indicating a lower internal query similarity. The distribution of random samples falls between the two. This result indicates that Low Diff Value data has higher internal consistency, while High Diff Value data tends to be more diversity.

Analysis Based on the analysis of the above four aspects, we can summarize the significant feature differences between Low Diff Value data and High Diff Value data:

Low Diff Value data tends to repeatedly use more common vocabulary to form longer token sequences, and has high internal consistency. In contrast, High Diff Value data contains more rare or special vocabulary and refuses to expand sequence length, resulting in higher diversity. These feature differences may explain why Low Diff Value data is crucial for model training performance while High Diff Value data becomes 'dirty data'. The uniqueness and inconsistency of High Diff Value data may introduce noise/contain outliers/incorrect labeling/information unrelated to the main task, thereby interfering with the model's learning of general patterns. On the contrary, the consistency and universality of Low Diff Value data help the model capture key features and common patterns, maintain model stability, avoid overfitting to small or abnormal samples, and thus improve generalization ability and overall performance.

6 Conclusion

In this paper, we introduced ResoFilter, a novel approach for refining fine-tuning datasets obtained through GPT-based data augmentation. Our experiments showed that ResoFilter can maintain or improve training performance while removing up to 50% of redundant data across various model architectures, sizes, and task domains. This work provides insights into balancing data richness and specificity, optimizing model performance and computational efficiency. ResoFilter enhances model training and opens new avenues for research in data annotation, augmentation, and domain-specific fine-tuning strategies, with potential applications in developing more robust and efficient language models.

Limitations

Despite the demonstrated efficacy of ResoFilter through empirical experimentation, several limitations and avenues for future research persist. Firstly, the methodology could be further refined through the implementation of a multi-indicator approach, potentially enhancing the granularity and precision of the data filtering process. Secondly, the insights derived from ResoFilter could be utilized to inform the reverse-engineering of dataset construction, potentially yielding more optimal training datasets ab initio. Finally, the current study is constrained by the absence of comprehensive evaluations on very large-scale models (e.g., those exceeding 70 billion parameters), alternative architectures such as Mixture of Experts (MoE), and specialized systems like conversational models. Addressing these limitations in subsequent research would provide a more exhaustive understanding of ResoFilter's applicability and impact across a diverse spectrum of model scales and architectures.

References

Yelaman Abdullin, Diego Molla-Aliod, Bahadorreza Ofoghi, John Yearwood, and Qingyang Li. 2024. Synthetic dialogue dataset generation using llm agents. *arXiv preprint arXiv:2401.17461*.

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

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: When data mining meets large language model finetuning. *Preprint*, arXiv:2307.06290.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, et al. 2023. Auggpt: Leveraging chatgpt for text data augmentation. *arXiv preprint arXiv:2302.13007*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Preprint*, arXiv:2305.00586.

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

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14409–14428, Toronto, Canada. Association for Computational Linguistics.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Po-Nien Kung, Fan Yin, Di Wu, Kai-Wei Chang, and Nanyun Peng. 2023. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. *arXiv preprint arXiv:2311.00288*.

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

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024a. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024b. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*.

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

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

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Junhao Liu, Tongliang Liu, Fei Huang, et al. 2023b. One shot learning as instruction data prospector for large language models. *arXiv preprint arXiv:2312.10302*.

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

Ziche Liu, Rui Ke, Feng Jiang, and Haizhou Li. 2024. Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models. *ArXiv*, abs/2406.14115.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kshitij Mishra, Tamer Soliman, Anil Ramakrishna, A. G. Galstyan, and Anoop Kumar. 2024. Correcting language model outputs by editing salient layers. In *Findings*.

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

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve

Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *NAACL*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Tobias Schimanski, Jingwei Ni, Mathias Kraus, Elliott Ash, and Markus Leippold. 2024. Towards faithful and robust llm specialists for evidence-based question-answering. *Preprint*, arXiv:2402.08277.

Kaiser Sun and Mark Dredze. 2024. Amuro & char: Analyzing the relationship between pre-training and fine-tuning of large language models. *Preprint*, arXiv:2408.06663.

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

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

The Mistral AI Team. 2024. Mistral-7B-v0.3. https://huggingface.co/mistralai/Mistral-7B-v0.3. Accessed: 2024-09-24.

I Tenney. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

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. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Lai Wei, Zihao Jiang, Weiran Huang, and Lichao Sun. 2023. Instructiongpt-4: A 200-instruction paradigm for fine-tuning minigpt-4. *Preprint*, arXiv:2308.12067.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. *Preprint*, arXiv:2402.04333.

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. *arXiv preprint arXiv:2304.12244*.

Zihan Yao, Yu He, Tianyu Qi, and Ming Li. 2024. Scalable model editing via customized expert networks. *ArXiv*, abs/2404.02699.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Training Data Format

When constructing the instruction fine-tuning data, we followed the formats: Gemma2, Phi2, and Mistral used one set of templates, while Llama2 used another set of templates. Taking the Math dataset as an example, the specific formats are as follows:

Gemma2,Phi2,Mistral

<start_of_turn>user
Calculate 8 divided by $\frac{1}{8}$.**<end_of_turn>**
<start_of_turn>model
Dividing by a fraction is the same as multiplying by its reciprocal.
So, $8 \div \frac{1}{8} = 8 \times \frac{8}{1} = 64$.
The answer is: 64 **<end_of_turn>**

Llama2

[INST]Dave bought 8 books about animals, 6 books about outer space, and 3 books about trains to keep him busy over the holidays. Each book cost 6. How much did Dave spend on the books?.**[/INST]**Dave bought a total of 8 + 6 + 3 = 17 books
Each book cost 6, so Dave spent a total of 17 x 6 = 102 on the books.
102
The answer is: 102 **</s>**

B Analysis of General Performance

Model	MMLU Score
gemma2-2b	0.5300
gemma2-2b-it	0.5605
dolly	0.4658
tulu	0.3733
mixed	0.4087
wizard	0.3184
flan	0.4064
flan+cot	0.4244

Table 5: Comparison of different models on MMLU.

Model	full	25%	50%	75%
dolly	0.3704	0.3861	0.3620	0.3778
flan cot	0.3148	0.3435	0.3056	0.2861
metamath	0.3565	0.3889	0.4000	0.3454
evol-codealpaca-v1	0.3769	0.3861	0.3824	0.3870

Table 6: Our evaluation results on the BBH.

To improve the model's performance on general tasks following fine-tuning, we experimented with a range of fine-tuning datasets. For the MMLU benchmark, we performed experiments with several training sets, as detailed in Table 5, and found it nearly impossible to match the performance of the instruction-tuned version of the Gemma2 model. Moreover, the MMLU metrics consistently declined after fine-tuning when compared to the original model. Additionally, as shown in Table 6, our experiments on BBH revealed that reducing the size of the training set often led to models that outperformed those fine-tuned on the full dataset. In certain datasets, smaller training sets even produced superior training outcomes. As noted by Sun and Dredze (2024), general model capabilities are inherently challenging to enhance through fine-tuning. Furthermore, with respect to the Gemma 2-2B model, it has exhibited exceptional performance given the limitations of its parameter scale, approaching the theoretical performance ceiling for models of similar size.

Module	25%	50%	75%
Random	0.6042	0.6254	0.6474
W_{up}	**0.6042**	**0.6497**	**0.6603**
W_{down}	0.589	0.652	0.6459
W_q	0.5891	0.6338	0.649
W_k	0.583	0.6444	0.6596
W_v	0.5588	0.6346	0.652

Table 7: Experimental results of various weight modules on GSM8k benchmark. W_{up} gets the best in three ratios. All weights show the same increasing trend performance.

Module	25%	50%	75%
Original Order	**0.6042**	**0.6497**	**0.6603**
Random Order	0.5913	0.636	0.6497
Min to Max	0.6065	0.6345	0.6436
Max to Min	0.5837	0.6322	0.6512

Table 8: Inspired by the concept of curriculum learning, we investigated the impact of data order on model performance. We calculated the mean difference scores for the W_{up} weight layer and filtered the data accordingly. The filtered data was then arranged in three ways: from highest to lowest score (Max to Min), from lowest to highest score (Min to Max), and randomly. We compared these arrangements with the original data order.

C Module Type

In most experiments we only use the W_{up} of the weight parameters in model to validate our ideas. Would the self attention weights contain more infomation to help to select data? The results in Table 7 showed that the data selected based on the W_{up} module achieved the best performance at all ratios, especially at data volumes of 50% and 75%, which were significantly better than the random baseline. Other modules, such as W_{down} and W_k, also perform well under certain data ratios. In contrast, the W_v module performs the worst at 25% data volume, but all the modules follow the same trends.

D Data Order

Inspired by the concept of curriculum learning, we explored the impact of data order on model performance. Table 8 presents the results of this experiment on the GSM8k benchmark. Interestingly, the original data order consistently outperformed the other arrangements across all percentages. The random order showed competitive performance, particularly at higher data percentages. The Min to Max order performed slightly better than Max to Min, especially at lower data percentages, suggesting that introducing easier examples first might be beneficial. However, the differences between these orderings were relatively small, indicating that the original data order already possesses an inherent structure that is conducive to effective learning.

E Algorithm

Algorithm 1 ResoFilter: Parameter Difference-based Data Filtering

Require: Initial model M_0, Training dataset D, Filtering ratio p
Ensure: Filtered dataset $D_{filtered}$

```
for each sample d_i ∈ D do
    Fine-tune M_0 using d_i to obtain M_i
    ΔW ← M_i − M_0 {Compute parameter difference}
    for each layer l ∈ M_i do
        for each module m ∈ l do
            diff_{l,m} ← mean(ΔW_{l,m})
        end for
    end for
    L_last ← {l_i | i ∈ [|M_i| − n + 1, |M_i|]} {Extract last n layers from M_i}
    diff_i ← mean({diff_{l,up_proj} | l ∈ L_last})
    Associate diff_i with d_i
end for
Sort D in descending order based on diff_i
k ← ⌊|D| × (1 − p)⌋ {Calculate number of samples to retain}
D_filtered ← Bottom k samples from sorted D, preserving original relative order {This effectively removes the top p% of samples}
return D_filtered
```

First, for each sample d_i in the dataset D, we fine-tune the initial model M_0 using d_i to obtain M_i. We then compute the parameter difference $\Delta W = M_i - M_0$, and for each layer and module in M_i, we calculate the average parameter difference. After conducting a series of experiments, we selected the W_{up} modules from the final n layers and computed their average differences, denoted as diff_i. This allows us to associate diff_i with the corresponding sample d_i. Next, we perform data filtering by sorting the dataset D in descending order based on diff_i. According to the desired number of retained samples k, we select the bottom k samples from the sorted dataset D to form D_{filtered}, while preserving their original relative order. The filtered dataset D_{filtered} is then provided directly to the model for the fine-tuning process.