Scaling Towards the Information Boundary of Instructions through Data Synthesizing

Li Du[*†], **Hanyu Zhao**[*], **Yiming Ju**[*], **Tengfei Pan**

Beijing Academy of Artificial Intelligence, Beijing, China
{hyzhao, duli, ymju, tfpan}@baai.ac.cn

Abstract

High-quality instructions are crucial for aligning pretrained models to improve their performance on downstream tasks. Although current instruction datasets have reached tens of millions of samples, models finetuned on them may still struggle with complex instruction following and tasks in rare domains. This is primarily due to limited expansion in both "coverage" (coverage of task types and knowledge areas) and "depth" (instruction complexity) of the instruction set. To address this issue, we propose a systematic instruction data construction framework, which integrates a hierarchical labeling system, an informative seed selection algorithm, an evolutionary data synthesis process, and a model deficiency diagnosis with targeted data generation. These components form an iterative closed-loop to continuously enhance the coverage and depth of instruction data. Based on this framework, we construct Infinity Instruct Subject, a high-quality dataset containing ∼1.5 million instructions. Experiments on multiple foundation models and benchmark tasks demonstrate its effectiveness in improving instruction-following capabilities. Further analyses suggest that Infinity Instruct Subject shows enlarged coverage and depth compared to comparable synthesized instruction datasets.

Code — https://github.com/BAAI-DIPL/InfinityInstruct-Sub

Dataset — https://huggingface.co/datasets/BAAI/Infinity-Instruct/tree/main/Gen

Introduction

Instruction tuning serves as a cornerstone for unleashing the vast knowledge and reasoning capabilities of large pretrained models (Longpre et al. 2023; Ouyang et al. 2022). Consequently, the construction of high-quality instruction datasets has become essential for improving model performance and generalization ability. Several efforts have constructed instruction datasets through manual annotation or automatic synthesis, with the total size of these datasets having reached tens of millions (Ahmad et al. 2025; Nayak et al. 2024). However, current large models still struggle with complex instruction-following tasks and show limitation in "rare" tasks with low frequency (Qin et al. 2025).

A main reason would be that existing works overlooked the distribution of existing data, and thus failed to efficiently enhance the "coverage and "depth of synthesized instructions, and thus limit the model performance in solving rare tasks and difficult tasks (Wang et al. 2024; Liu et al. 2024; Long et al. 2024). The "coverage of an instruction dataset refers to the coverage of task types and knowledge categories it covers, while the 'depth reflects the complexity of instructions within individual tasks. Both two dimensions are essential to enlarge the capability boundaries of large models (Shengyu et al. 2023). Expanding the coverage improves the model's ability to generalize across domains, while increasing depth helps enhance complex reasoning ability and compositional generalizability (Sinha, Premsri, and Kordjamshidi 2024; Zhong, Zhou, and Wang 2025). However, a key challenge remains: how to design **effective, controllable and interpretable** instruction generation strategies that continuously expand the instruction space along both dimensions, guiding models to acquire higher-level capabilities (Zhang et al. 2023; Qian et al. 2022; Qin et al. 2022). Currently, there is a lack of unified methodology and empirical research in this area.

To address this challenge, we propose a comprehensive instruction data construction framework, which goal is to measure the current data distribution and synthesize instructions that target under-covered and high-complexity regions, thereby expanding both the coverage and depth of the dataset. This framework consists of four core components: (1) a hierarchical and multilingual tagging system to understand the content and ability distribution of existing instruction content; (2) informative seed data selection to identify valuable instructions with low coverage or high difficulty; (3) evolutionary data synthesis to synthesize more complex instructions by evolving from seed data; (4) a model deficiency diagnosis based targeted synthesis module to identify potential flaws in the model's knowledge or capabilities and synthesize data to address those weaknesses. As illustrated in Figure 1, these four modules form a closed-loop system that can iteratively expand the coverage and depth of the instruction dataset.

Based on this framework, we construct the Infinity Instruct Subject (InfInstruct-Sub) dataset, which contains

[*]Equal contribution.
[†]Corresponding Author

$\sim$1.5 million high-quality instructions. The code and dataset are publicly released. Evaluations on multiple benchmarks demonstrate its effectiveness in improving instruction-following ability. With finetuning on InfInstruct-Sub, different series of foundation models show further performance improvements compared to their official instruct models, and certain versions of GPT4. Data analysis further suggests that, compared to synthetic datasets such as Magpie (Wei et al. 2023) and UltraChat(Ding et al. 2023), our framework could obtain an instruction set with higher coverage and complexity.

Construction of Infinity Instruct Subject

The construction process of Infinity Instruct Subject (InfInstruct-Sub) is illustrated in Figure 1. The process starts by collecting high-quality seed instructions. We systematically gathered all general-domain instruction datasets both manually created and automatically generated using GPT-4(Achiam et al. 2023) or ChatGPT available up to March 2024, resulting in a data pool of approximately 7 million samples. The included datasets are listed in Table **??** of the Appendix. Next, we developed an automatic tagging system based on large language models (LLMs) to analyze the distribution of the data pool. From the perspectives of coverage and depth, we selected a set of high-information seed instructions. Then, we applied an evolutionary algorithm to generate over one million new instruction samples by evolving towards greater complexity and difficulty. Building on both the seed and synthesized data, we constructed a model deficiency diagnosis system. This system identifies gaps in model capabilities and guides the targeted synthesis of new data to efficiently address those weaknesses. Finally, we implemented a strict semantic similarity–based data leakage prevention framework to detect and mitigate potential risks of data leakage throughout the construction process.

Hierarchical Multilingual Tagging System for Characterizing Instruction Content Distribution

A prerequisite for synthesizing instructions that target under-covered areas or exhibit high complexity is understanding the content distribution of existing instruction data. To this end, we design a hierarchical, multilingual tagging system powered by LLMs as illustrated in Figure 2. This system assigns each instruction both Chinese and English tags at two levels: domain-level tags and fine-grained tags. These tags indicate the domain of each instruction, as well as the types of abilities and knowledge required to complete it, facilitating interpretation and selection for downstream tasks.

Specifically, the tagging process is implemented in a bottom-up manner. We first generate fine-grained tags for each instruction. These are then normalized (Figure 2(a)) to remove noise and merge semantically equivalent tags expressed in different forms. Finally, LLMs are used to automatically cluster and abstract fine-grained tags into broader domain-level tags, and to map each fine-grained tag to its corresponding domain tag (Figure 2(b)). The reason for adopting such an automatic bottom-up manner is that, due to the complex interrelated and vast scope of the skills and abilities of LLMs, it would be rather impractical to categorize them into an orthogonal taxonomy manually.

Fine-grained Tagging System Specifically, given an instance from the instruction dataset consisting of one or more Query–Response pairs, we concatenate them into a single string. We then use a carefully designed prompt to aviod generating over-detailed or over-coarse tags, as illustrated in Figure **??**, to instruct LLMs to generate tags that describe the knowledge and skills required to complete the dialogue[1].

Tag Normalization Since LLMs may describe the same knowledge or skill using different expressions (e.g., "math calculation" vs. "mathematical calculation"), we normalize tags based on semantic similarity. Specifically, we obtain embeddings for all tags using BGE (Chen et al. 2024), and identify semantically similar tags whose cosine similarity exceeds an empirical threshold of $\lambda = 0.91$. Within each group of similar tags, we retain the one with the highest frequency as the representative. To further refine the tag set, we apply DBSCAN (Ester et al. 1996) clustering with parameters $eps = 0.47$ and $min\_samples = 2$ to merge closely related tags. These thresholds are chosen based on empirical observations. After normalization and clustering, tags with frequency less than 100 are considered noisy and removed, following prior work (Lu et al. 2023). As a result, a total of 21,378 fine-grained tags are retained.

Domain Tagging System Given the set of fine-grained tags, we select the top 1,000 most frequent ones, then use GPT-4 to automatically summarize them into a set of domain-level categories, so that we employ each induced category as a domain-level tag. To establish a mapping between fine-grained tags and domain-level tags, we use the prompt shown in Figure **??** along with the categorization criteria illustrated in Figure **??** of the Appendix. Based on this setup, we employ Qwen2.5-72B-Instruct (Yang, Yang, and Zhang 2024) to generate the mappings between fine-grained and domain tags.

Informative Seed Instructions Selection

The initial data distribution largely determines the effectiveness of subsequent data synthesis, thus selecting a set of informative seed instructions is crucial for synthesizing high-quality instruction data. Given the abundance of existing instruction datasets, aimlessly synthesizing existing data is of limited value. In contrast, it is more meaningful to focus on instructions that are currently underrepresented in existing datasets (expanding coverage), or those that are relatively difficult and expose model limitations in reasoning or knowledge (enhancing depth)(Li et al. 2024a; Shen et al. 2025; Huang et al. 2025; Sun et al. 2024). We design four criteria to select informative instructions:

1. Hard-to-Follow Instructions: Inspired by (Li et al. 2023a; Qin et al. 2024), we select the 50k instructions with the smallest reduction in token-level average cross-entropy

[1]We use Qwen-2.5-72B-Instruct as the tagging model to ensure high-quality annotations.

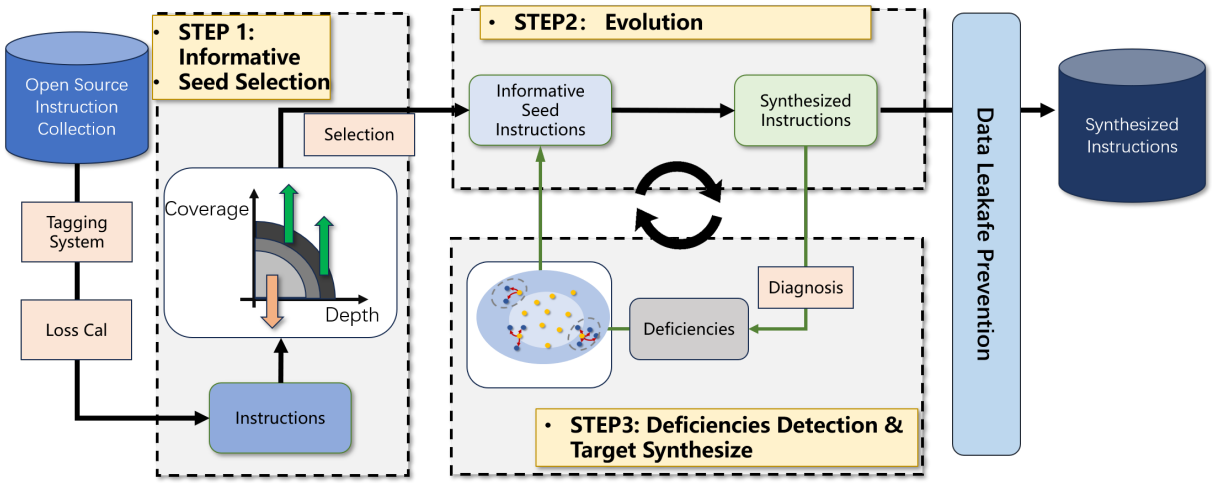

Figure 1: Construction pipeline of the InfInstruct-Sub dataset.

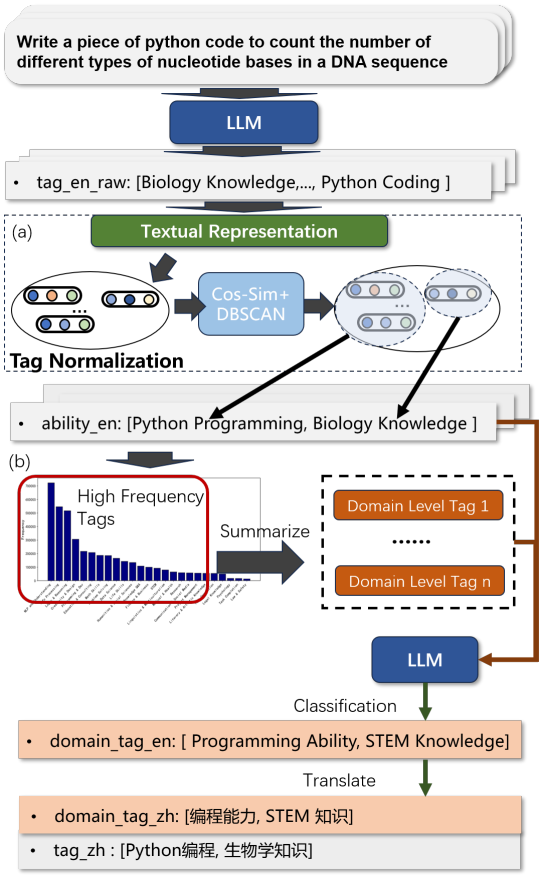

Figure 2: Tagging system of the InfInstruct-Sub dataset for elucidating content distribution of instruction pools. (a) Fine-grained tags and normalization of fine-grained tags. (b) Construction of categorical tags and the process of mapping fine-grained tags to categorical tags.

loss after fine-tuning, ensuring the retained instructions are challenging to follow.

2. Long-Tailed Instructions: To guarantee coverage, we include instructions that contain at least one fine-grained tag with a frequency below 200 in the seed set. Additionally, we randomly sample 30% of the instructions that contain tags with frequencies in the range [200, 500].

3. Multi-Skill Required Complex Instructions: Instructions associated with more than four fine-grained tags are included, as such instructions require more comprehensive reasoning or knowledge.

4. Undertrained Instructions:

These are instructions for which the base model performs poorly with a high loss value. This may either due to either a lack of exposure or inherent difficulty. We include 200k instructions with loss greater than mean(loss) $+ 1.96 \times$ std(loss).

In particular, we use Llama-2-7B-base (Touvron et al. 2023) to compute the loss values for instructions. To estimate the reduction ratio in token-level average cross-entropy loss, we fine-tune Llama-2-7B-base on a balanced instruction subset with 1,000 instructions per domain category. Table **??** presents the sources of the seed data and the number of instructions retained after filtering. By applying the above criteria sequentially, we select approximately 1.2 million informative instructions from the overall data pool.

Evolutionary Algorithm for In-Distribution Generalization

Based on the previously selected informative instructions as seeds, we employ an evolutionary algorithm to synthesize new instructions, so as to ensure the performance on instruction sets similar to the seed instruction distribution, meanwhile enlarge the coverage and increasing the depth of seed instructions. Specifically, we design a three-step evolution process:

1. Seed-based Evolution: In this step, we adopt the widely-adopted Evol-Instruct (Xu et al. 2023) method. By

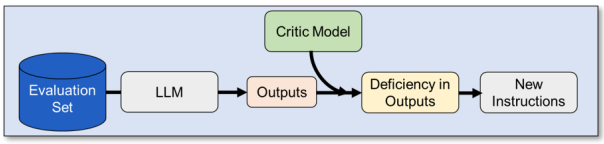

Figure 3: Illustration of the deficiency diagnosis process.

randomly choose one of four dimensions—diversity, i.e., more reasoning steps, concretizing, deepening, and more diversified, to guide each instruction's evolution using State-of-the-Art advanced large model as synthesizer.

2. Validation and Filtering: Following (Törnberg 2023), evolved instructions are evaluated by an advanced large model to judge the quality, as well as the similarity between the original seed instruction.

3. Multi-Turn Dialogue Generation: Following (Ding et al. 2023), each valid instruction is used to generate 1–4 rounds of dialogue, with the AI assistant simulating different roles.

Deficiency Diagnosis and Defect-Driven Instruction Synthesis

Despite comprehensive data collection efforts, many topics remain underrepresented, and the model may still fail to capture key knowledge and skills within the instruction set, leading to capability and knowledge deficiency. While expanding the seed set or synthesizing more data via evolutionary methods can help, these approaches often produce redundant and inefficient results. To address this issue, we adopt a more targeted strategy: directly diagnosing model deficiencies and synthesizing instructions specifically to fill those gaps.

As illustrated in Figure 3, we first construct a diagnosis dataset $\mathcal{D} = \mathcal{D}_{i=1}^{N}$ drawn from the same distribution as the seed dataset, where each $\mathcal{D}_i = \{x_i, y_i\}$ represents an input query and its corresponding reference response. We then use the fine-tuned model M_{FT} to generate a response $\hat{y}_i$ for each x_i. Next, we compare $\hat{y}_i$ with y_i using a State-of-the-Art advanced large Oracle model, identifying knowledge or deficiencies in $\hat{y}_i$ with the prompt shown in Figure 2 of the Appendix. Given the diagnosed issues, we then prompt the Oracle model to generate new instructions that specifically address these deficiencies using the prompt in Figure 3 of the Appendix, to target generate instructions that remedy deficiencies within the model.

Data Leakage Prevention

Overlap between training and evaluation data can cause models to achieve inflated scores on familiar samples, leading to overestimated performance. We observe that many public instruction datasets include questions from common benchmarks like MT-Bench (Zheng et al. 2023b) and AlpacaEval (Li et al. 2023b), which risks misleading evaluation. To address this issue, we use the BGE model to compute semantic similarity between evaluation queries and instructions in the training data. This process involves multi-dimensional comparisons, including semantic understanding and structural features, to ensure accurate and comprehensive matching. We filter out all synthesized instructions with a similarity score larger than 0.45 to any samples within MT-Bench and AlpacaEval, to prevent potential data leakage and preserve the reliability of model evaluation.

After the above procedures, we obtain 146,9391 new instruction samples.

Performance Analysis

Experimental Setup

We finetune open-source pretrained models Qwen2-7B-base and LLaMA3-8B-base on the InfInstruct-Sub dataset, and compare their performance against their respective official instruction-tuned and alignment-tuned versions. Furthermore, we conduct a systematic comparison on LLaMA3-8B-base across ten prominent instruction-tuning datasets—Self-Instruct(Wang et al. 2023), ShareGPT(Zheng et al. 2023b), Evol-Instruct(Xu et al. 2023) , OpenHermes-1(Teknium 2023a), Tulu-V2-Mix(Ivison et al. 2023) , WildChat(Zhao et al. 2024a) , OpenHermes-2.5(Teknium 2023b), GenQA(University of Maryland, College Park 2024), UltraChat(Ding et al. 2023), and Magpie(Xu et al. 2024) —to comprehensively evaluate their performance differences. Evaluations are conducted on widely adopted LLM-judging-based benchmarks AlpacaEval 2.0 (Li et al. 2023b) and Arena-Hard-V0.1 (Li et al. 2024b). The reference model for AlpacaEval 2.0 is GPT-4-1106; on Arena-Hard-V0.1 is GPT-4-0314. More details about hyperparameters are provided in the Appendix.

Experimental Results

Table 1 shows the performance of base models Llama3-8B, Qwen-2-7B, and Mistral-7B-V0.2 (abbreviated as Mist.V0.2) fine-tuned with InfInstruct-Sub on AlpacaEval 2.0 and Arena-Hard-V0.1. Compared to earlier and concurrent instruction datasets, models fine-tuned with InfInstruct-Sub show improved performance, especially on ArenaHard, which focuses on more complex tasks. Moreover, comparison with performance using instruction collection OpenHermes 2.5 (**?**) and GenQA (University of Maryland, College Park 2024), which have similar or larger sizes, shows that simply enlarging the size of the instruction set would not necessarily lead to performance improvement, particularly for harder instructions. These highlight the necessity of enhancing the depth of the instruction set. Moreover, on Llama, Qwen, and Mistral series of model, the performance of models finetuned upon InfInstruct-Sub outperforms official instruction-tuned counterparts, and surpasses earlier versions of GPT4, using the 7B-sized Mistral-V0.2 base model. Notably, InfInstruct-Sub applies strict data leakage prevention and offers high coverage and difficulty, enabling improved generalization across diverse tasks. These results demonstrate the effectiveness of our proposed data construction framework to enhance model performance and its general applicability in enhancing instruction-following performance across different base models.

To further validate the effectiveness of the seed selection process, we randomly sampled 5,0000 instances from the

Model	Datasize	AlpacaE.	AH
GPT-4-1106	–	50.0	–
GPT-4-0314	–	35.3	50.0
GPT-4-0613	–	30.2	37.9
Llama-3-8B			
+Self-Instruct	100k	7.2	4.0
+ShareGPT	112k	9.7	6.5
+Evol Instruct	143k	8.5	5.1
+OpenHermes 1	243K	9.9	4.4
+Tulu V2 Mix	326K	9.9	5.4
+WildChat	652k	14.6	8.7
+OpenHermes 2.5	1M	12.9	8.2
+GenQA	6.47M	9.1	3.0
+UltraChat	208k	8.3	3.6
+Magipie	300k	22.7	14.9
Llama-3-8B-Instruct	>10M	22.9	20.6
Llama-3-8B+InfInstruct-Sub	1.46M	**36.2**	**35.3**
Qwen-2-7B-Instruct	-	20.9	19.6
Qwen-2-7B+InfInstruct-Sub	1.46M	**28.1**	**27.7**
Mist.V0.2-7B-Instruct	-	17.1	16.2
Mist.V0.2-7B+InfInstruct-Sub	1.46M	**41.2**	**38.0**

Table 1: Performance comparison of various models across different benchmarks. AH. is the abbr. of ArenaHard.

	AplacaEval	AH.
Llama3-8B + Ori_Random	4.5	3.5
Llama3-8B + Seed	6.4	3.9

Table 2: Performance of Llama3-8B finetuned on data sampled from the original instruction pool and seed instructions. AH. is the abbr. of ArenaHard.

original instruction collection, and the selected seed instruction set, respectively, then finetuned Llama3-8B on these two instruction sets. As shown in Table 2, base model finetuned with seed instructions show improved performance on both benchmarks. This suggests the effectiveness of the seed selection process, and the necessity of synthesizing high-quality instructions, due to the limited performance upon open-source instructions.

Dataset Analysis

Dataset Distribution

Figure 4 shows the distribution of InfInstruct-Sub instructions across different domain labels. Figure 5 visualizes the instruction data projected into semantic space using BGE as the text encoder, followed by t-SNE for dimensionality reduction. Overall, instructions with different domain labels are distributed in clearly separated regions, suggesting that instructions associated with different semantic tags carry distinct meanings. This supports the reasonability of our tagging system.

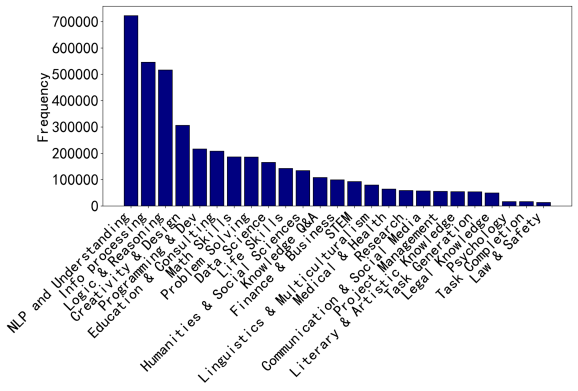

Figure 4: Illustration of the domain category distribution of the InfInstruct-Sub dataset.

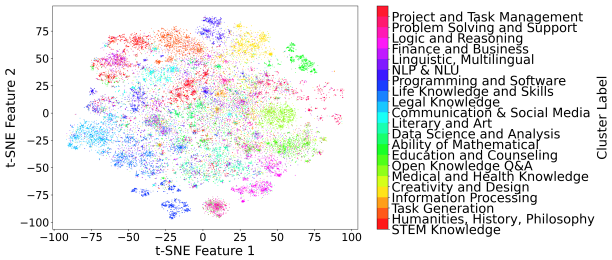

Figure 5: Illustration of the distribution within the semantic space of the synthesized instructions.

Distribution within the semantic space We further compare the semantic coverage of InfInstruct-Sub with comparable instruction datasets, including AlpacaGPT4(Peng et al. 2023), LLM-Sys(Zheng et al. 2023a), and Magpie (Xu et al. 2024). We sample 20,000 instructions from each dataset and apply the embedding and projection method same to abovementioned. As shown in Figure 6, InfInstruct-Sub exhibits a broader spread in the semantic space, demonstrating that our strategy of selecting more low-frequency instructions as seeds effectively increases coverage in the synthesized data.

To further quantify the distribution diversity of instruction sets within the semantic space, we use spatial entropy (SE) to evaluate the distribution uniformity of each dataset. Spatial entropy is an extension of information entropy that quantifies the dispersion of points in space (Celik 2014). Higher spatial entropy indicates a more uniform and diverse distribution. Formally, we uniformly divide the 2D semantic space into n discrete grids and compute the probability p_i of a sample falling into the i-th grid. The spatial entropy is defined as:

$$H_{sp} = \sum p_i \log p_i \quad (1)$$

In practice, p_i is estimated as the proportion of samples falling into each grid cell. We divide the 2D space into a 200×200 grid and compute spatial entropy for each dataset accordingly. As shown in Table 3, Infinity Instruct Subject achieves higher spatial entropy, indicating a more uniform

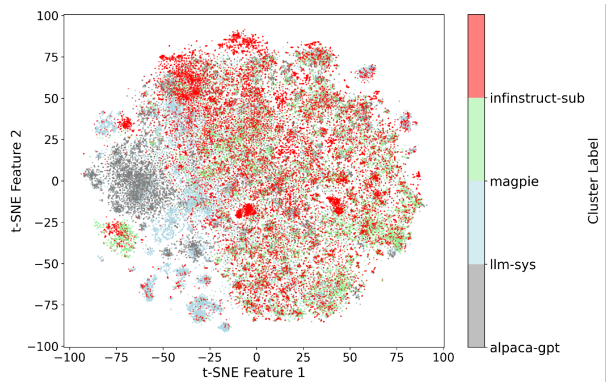

Figure 6: Distribution of the different instruction sets within the semantic space.

	AlpacaGPT4	LLM-Sys	Magipie	InfInstruct-Sub
SE	4.366	4.649	4.978	**5.023**

Table 3: Spatial entropy (SE) of instructions within the 2D-semantic space.

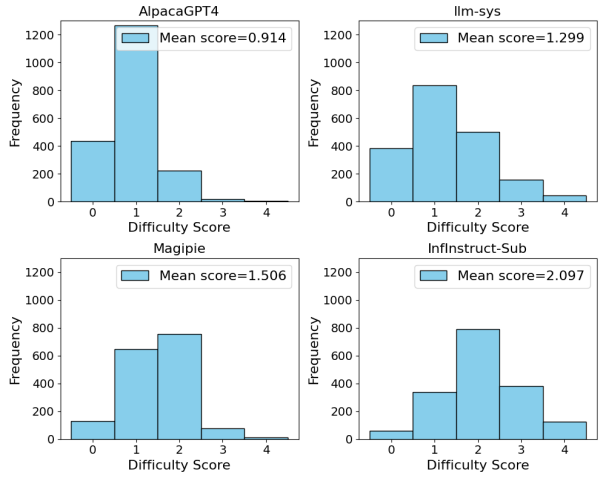

Figure 7: Distribution of difficulty scores of different instruction sets.

and diverse distribution. This results from our targeted seed selection strategy, which emphasizes low-frequency and high-difficulty instructions, as well as instructions where the model underperforms, thus avoiding redundancy and enhancing the coverage and diversity of dataset.

Difficulty of synthesized data To evaluate the difficulty of synthesized data, we follow the approach proposed in Magpie (Xu et al. 2024), using a large language model to assign difficulty scores to instruction samples. The difficulty is rated on a five-point scale: 'very easy': 0, 'easy': 1, 'medium': 2, 'hard': 3, 'very hard': 4. This evaluation process is independent of the method we use for selecting high-difficulty seed instructions. Specifically, we employ Qwen-2.5-32B-Instruct as the scorer and randomly sample 1,500 instructions from each of the following datasets: Alpaca-GPT4, LLM-Sys, Magpie, and Infinity Instruct Subject.

Figure 7 shows the distribution of difficulty scores and the average score for each dataset. Compared to others, Infinity Instruct Subject contains more high-difficulty instructions and achieves a higher average difficulty score. This helps explain the superior performance of models fine-tuned InfInstruct-Sub on challenging benchmarks such as Arena-Hard (Table 1). It also validates the effectiveness of our seed selection strategy in enhancing dataset difficulty through targeted depth-oriented instruction synthesis.

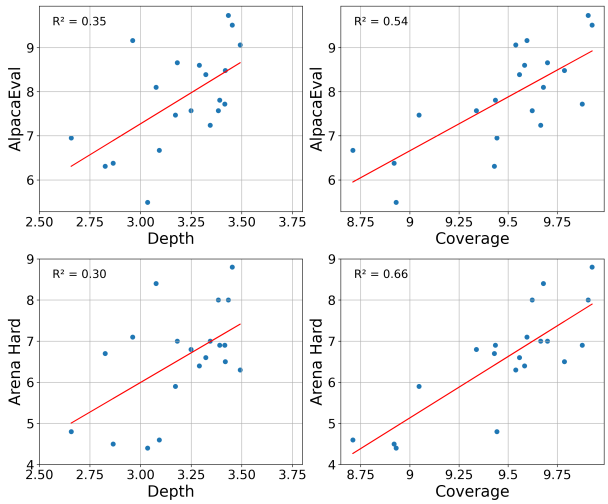

Figure 8: Performance of model finetuned with different coverage and depth.

Performance Scaling with Depth and Coverage

To further validate the effectiveness of our proposed method, we construct a series of instruction subsets from the synthesized data, each containing the same number of samples (20,000) but differing in depth and coverage. We define depth as the product of the logarithm of the instruction's label count and its token-level log loss of the base model. Coverage is measured by the logarithm of the number of non-empty grid cells occupied in the 2D semantic space described earlier.

Then, We fine-tune Llama3-8B on each subset and evaluate the aligned models on AlpacaEval and Arena-Hard. As shown in Figure 8, model performance increases consistently with instruction depth and coverage, even when dataset size is fixed. Intuitively, deeper instructions carry more informative signals, enabling the model to generalize better—including to simpler tasks. Previous studies have shown that instructions from different domains are difficult to substitute for each other (Zhao et al. 2025, 2024b). Therefore, it is necessary to enhance the coverage of instruction datasets to ensure model performance across various tasks. The correlation coefficient R^2 indicates that when the scale of the instruction dataset is around the tens of thousands, the model's performance may be more sensitive to the cover-

age of instruction coverage than to the depth of information. These findings highlight the importance of expanding both depth and coverage of the instruction set, and further support the effectiveness of our strategy in achieving this goal during instruction synthesis, to continuously enhance the model performance.

Related Work

Instruction Data Synthesis

Instruction tuning adapts base models to downstream tasks, with performance largely determined by the distribution of instruction data. Manual datasets such as LIMA (Zhou et al. 2023) and Dolly (Ding et al. 2023) provide high-quality expert-written instructions but are costly to scale, while semi-automatic approaches (e.g., Self-Instruct (Wang et al. 2023), Alpaca (Taori et al. 2023), Evol-Instruct (Muennighoff et al. 2023)) improve scalability via prompt-based expansion yet suffer from limited diversity due to handcrafted templates. Fully automatic pipelines such as WebInstruct (Yue et al. 2024) mine instruction-like data from the web but often lack precise control over coverage and difficulty.

Recent advances improve automatic instruction construction along three axes: (i) seed selection and filtering, where hierarchical tagging and multi-dimensional metrics identify high-information seeds beyond frequent and simple cases (Wang et al. 2023; Tu et al. 2024; He et al. 2025; Xie et al. 2023); (ii) evolution-based generation, which iteratively increases reasoning complexity through genetic or feedback-driven strategies (Xu et al. 2023; Gu et al. 2025); and (iii) instruction synthesis, where prompt-free or web-grounded methods enhance fluency and realism compared to prompt-based generation (Wang et al. 2023; Xu et al. 2023, 2024; Jiang et al. 2025).

Nonetheless, existing datasets still underrepresent complex, multi-step, and domain-specific instructions, limiting generalization to reasoning-intensive or specialized tasks (Yu et al. 2024; Zhang et al. 2025), and few works unify seed selection, feedback, and evolution in a closed-loop framework (Le et al. 2022). To address this gap, we propose a unified instruction data framework that jointly optimizes coverage and complexity via hierarchical labeling, high-information filtering, evolution-based synthesis, and deficiency-driven augmentation.

Model Self-Improvement

Model self-improvement aims to iteratively enhance model capabilities through self-generated data or feedback signals. Representative approaches include self-refinement (Madaan et al. 2023; Zhou et al. 2024), multi-round generation and evaluation loops, and performance-driven data augmentation (Yang et al. 2025). These methods enable models to identify and address their own weaknesses by generating more challenging training samples, thereby facilitating continual improvement.

To ensure data quality, recent studies introduce deficiency diagnosis mechanisms (Lighterness et al. 2024), which analyze model performance on downstream tasks to detect knowledge gaps or skill deficiencies. These insights guide the targeted synthesis of training data(Miller et al. 2025). By incorporating such feedback into the training loop, models and datasets can co-evolve—improving both model performance and data coverage over time(Bauer et al. 2024).

Beyond simply increasing instruction quantity, recent research emphasizes instruction coverage (task/domain diversity) and depth (reasoning complexity). Several works have explored combining hierarchical tagging with complexity control to synthesize increasingly challenging and diverse instructions via evolutionary algorithms (Lu et al. 2023; Zhao et al. 2025). These strategies enhance the cognitive depth of training data and promote models' abilities in complex reasoning and compositional generalization.

Conclusion

In this paper, we propose a novel framework for constructing high-quality instructions that continuously expand both the coverage and complexity of instruction data. By combining a hierarchical multilingual tagging system, informative seed instruction selection, evolutionary data synthesis, and model deficiency diagnosis, our framework enables generating instructions that effectively address gaps in coverage and complexity within existing instruction datasets. We show the effectiveness of our framework through the Infinity Instruct Subject dataset containing over 1.5 million high-quality instructions, and show that it improves the instruction-following capabilities of foundation models across benchmarks. Further analyses show that our methodology can obtain instructions with enhanced coverage and depth.

Acknowledgements

We thank the support of the National Science and Technology Major Project (2022ZD0116301), and the Youth Fund of the National Natural Science Foundation of China (62406040).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ahmad, W. U.; Ficek, A.; Samadi, M.; Huang, J.; Noroozi, V.; Majumdar, S.; and Ginsburg, B. 2025. OpenCodeInstruct: A Large-scale Instruction Tuning Dataset for Code LLMs. *arXiv preprint arXiv:2504.04030*.

Bauer, K.; Heigl, R.; Hinz, O.; and Kosfeld, M. 2024. Feedback loops in machine learning: A study on the interplay of continuous updating and human discrimination. *Journal of the Association for Information Systems*, 25(4): 804–866.

Celik, T. 2014. Spatial entropy-based global and local image contrast enhancement. *IEEE Transactions on Image Processing*, 23(12): 5298–5308.

Chen, J.; Xiao, S.; Zhang, P.; Luo, K.; Lian, D.; and Liu, Z. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings

through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Ding, N.; Chen, Y.; Xu, B.; Qin, Y.; Hu, S.; Liu, Z.; Sun, M.; and Zhou, B. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. In *Proceedings of the EMNLP 2023*, 3029–3051.

Ester, M.; Kriegel, H.-P.; Sander, J.; and Xu, X. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second KDD*, 226–231. AAAI Press.

Gu, S.; Liu, J.; Li, D.; Zhang, G.; Han, M.; Gu, H.; Zhang, P.; Gu, N.; Shang, L.; and Lu, T. 2025. LLM-Based User Simulation for Low-Knowledge Shilling Attacks on Recommender Systems. *arXiv preprint arXiv:2505.13528*.

He, J.; Fan, Z.; Kuang, S.; Xiaoqing, L.; Song, K.; Zhou, Y.; and Qiu, X. 2025. FiNE: Filtering and Improving Noisy Data Elaborately with Large Language Models. In *Proceedings of the 2025 NAACL: Human Language Technologies (Volume 1: Long Papers)*, 8686–8707.

Huang, H.; Liu, J.; He, Y.; Li, S.; Xu, B.; Zhu, C.; Yang, M.; and Zhao, T. 2025. MuSC: Improving Complex Instruction Following with Multi-granularity Self-Contrastive Training. *arXiv e-prints*, arXiv–2502.

Ivison, H.; Wang, Y.; Pyatkin, V.; Lambert, N.; Peters, M. E.; Dasigi, P.; Jang, J.; Wadden, D.; Smith, N. A.; Beltagy, I.; et al. 2023. Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2. *CoRR*.

Jiang, Y.; Wang, Y.; Wu, C.; Dai, X.; Xu, Y.; Gan, W.; Wang, Y.; Jiang, X.; Shang, L.; Tang, R.; et al. 2025. Instruction-Tuning Data Synthesis from Scratch via Web Reconstruction. *arXiv preprint arXiv:2504.15573*.

Le, H.; Wang, Y.; Gotmare, A. D.; Savarese, S.; and Hoi, S. C. H. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in NIPS*, 35: 21314–21328.

Li, M.; Zhang, Y.; Li, Z.; Chen, J.; Chen, L.; Cheng, N.; Wang, J.; Zhou, T.; and Xiao, J. 2023a. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*.

Li, M.; Zhang, Y.; Li, Z.; Chen, J.; Chen, L.; Cheng, N.; Wang, J.; Zhou, T.; and Xiao, J. 2024a. From Quantity to Quality: Boosting LLM Performance with Self-Guided Data Selection for Instruction Tuning. In *NAACL 2024*, 7595–7628.

Li, T.; Chiang, W.-L.; Frick, E.; Dunlap, L.; Zhu, B.; Gonzalez, J. E.; and Stoica, I. 2024b. From live data to high-quality benchmarks: The arena-hard pipeline. *lmsys Blog.(Apr. 19, 2024),[Online]. Available: https://lmsys. org/blog/2024-04-19-arena-hard/(visited on 08/04/2024)*.

Li, X.; Zhang, T.; Dubois, Y.; Taori, R.; Gulrajani, I.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023b. Alpacaeval: An automatic evaluator of instruction-following models.

Lighterness, A.; Adcock, M.; Scanlon, L. A.; and Price, G. 2024. Data Quality–Driven Improvement in Health Care: Systematic Literature Review. *Journal of Medical Internet Research*, 26: e57615.

Liu, R.; Wei, J.; Liu, F.; Si, C.; Zhang, Y.; Rao, J.; Zheng, S.; Peng, D.; Yang, D.; Zhou, D.; et al. 2024. Best practices and lessons learned on synthetic data. *arXiv preprint arXiv:2404.07503*.

Long, L.; Wang, R.; Xiao, R.; Zhao, J.; Ding, X.; Chen, G.; and Wang, H. 2024. On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. In *Findings of the ACL 2024*, 11065–11082.

Longpre, S.; Hou, L.; Vu, T.; Webson, A.; Chung, H. W.; Tay, Y.; Zhou, D.; Le, Q. V.; Zoph, B.; Wei, J.; et al. 2023. The flan collection: designing data and methods for effective instruction tuning. In *ICML 2023*, 22631–22648.

Lu, K.; Yuan, H.; Yuan, Z.; Lin, R.; Lin, J.; Tan, C.; Zhou, C.; and Zhou, J. 2023. InsTag: Instruction Tagging for Analyzing Supervised Fine-tuning of Large Language Models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in NIPS*, 36: 46534–46594.

Miller, R.; Chan, S. H. M.; Whelan, H.; and Gregório, J. 2025. A Comparison of Data Quality Frameworks: A Review. *Big Data and Cognitive Computing*, 9(4): 93.

Muennighoff, N.; Liu, Q.; Zebaze, A.; Zheng, Q.; Hui, B.; Zhuo, T. Y.; Singh, S.; Tang, X.; Von Werra, L.; and Longpre, S. 2023. Octopack: Instruction tuning code large language models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Nayak, N.; Nan, Y.; Trost, A.; and Bach, S. 2024. Learning to Generate Instruction Tuning Datasets for Zero-Shot Task Adaptation. In *Findings of the ACL 2024*, 12585–12611.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in NIPS*, 35: 27730–27744.

Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J. 2023. Instruction Tuning with GPT-4. *arXiv preprint arXiv:2304.03277*.

Qian, J.; Dong, L.; Shen, Y.; Wei, F.; and Chen, W. 2022. Controllable Natural Language Generation with Contrastive Prefixes. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2912–2924.

Qin, L.; Welleck, S.; Khashabi, D.; and Choi, Y. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in NIPS*, 35: 9538–9551.

Qin, Y.; Li, G.; Li, Z.; Xu, Z.; Shi, Y.; Lin, Z.; Cui, X.; Li, K.; and Sun, X. 2025. Incentivizing Reasoning for Advanced Instruction-Following of Large Language Models. *arXiv preprint arXiv:2506.01413*.

Qin, Y.; Song, K.; Hu, Y.; Yao, W.; Cho, S.; Wang, X.; Wu, X.; Liu, F.; Liu, P.; and Yu, D. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Shen, S.; Huang, F.; Zhao, Z.; Liu, C.; Zheng, T.; and Zhu, D. 2025. Long Is More Important Than Difficult for Training Reasoning Models. *CoRR*.

Shengyu, Z.; Linfeng, D.; Xiaoya, L.; Sen, Z.; Xiaofei, S.; Shuhe, W.; Jiwei, L.; Hu, R.; Tianwei, Z.; Wu, F.; et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Sinha, S.; Premsri, T.; and Kordjamshidi, P. 2024. A survey on compositional learning of ai models: Theoretical and experimetnal practices. *arXiv preprint arXiv:2406.08787*.

Sun, H.; Liu, L.; Li, J.; Wang, F.; Dong, B.; Lin, R.; and Huang, R. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford alpaca: An instruction-following llama model.

Teknium. 2023a. OpenHermes 1. https://huggingface.co/datasets/teknium/OpenHermes. Accessed: 2024-09-08.

Teknium. 2023b. OpenHermes 2.5: An Open Dataset of Synthetic Data for Generalist LLM Assistants. https://huggingface.co/datasets/teknium/OpenHermes-2.5. Accessed: 2024-09-08.

Törnberg, P. 2023. Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning. *arXiv preprint arXiv:2304.06588*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tu, Z.; Meng, X.; He, Y.; Yao, Z.; Qi, T.; Liu, J.; and Li, M. 2024. ResoFilter: Rine-grained Synthetic Data Filtering for Large Language Models through Data-Parameter Resonance Analysis. *arXiv e-prints*, arXiv–2412.

University of Maryland, College Park. 2024. GenQA Dataset. https://github.com/UMD-COMP-LING/GenQA. Accessed: 2024-11-14.

Wang, K.; Zhu, J.; Ren, M.; Liu, Z.; Li, S.; Zhang, Z.; Zhang, C.; Wu, X.; Zhan, Q.; Liu, Q.; et al. 2024. A survey on data synthesis and augmentation for large language models. *arXiv preprint arXiv:2410.12896*.

Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *ACL*.

Wei, Y.; Wang, Z.; Liu, J.; Ding, Y.; and Zhang, L. 2023. Magicoder: Source Code Is All You Need. arXiv:2312.02120.

Xie, S. M.; Santurkar, S.; Ma, T.; and Liang, P. S. 2023. Data selection for language models via importance resampling. *Advances in NIPS*, 36: 34201–34227.

Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; and Jiang, D. 2023. WizardLM: Empowering Large Language Models to Follow Complex Instructions. *arXiv e-prints*, arXiv–2304.

Xu, Z.; Jiang, F.; Niu, L.; Deng, Y.; Poovendran, R.; Choi, Y.; and Lin, B. Y. 2024. Magpie: Alignment Data Synthesis from Scratch by Prompting Aligned LLMs with Nothing. *CoRR*.

Yang, A.; Yang, B.; and Zhang, B. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Yang, D.; Zeng, L.; Rao, J.; and Zhang, Y. 2025. Knowing You Don't Know: Learning When to Continue Search in Multi-round RAG through Self-Practicing. *arXiv e-prints*, arXiv–2505.

Yu, H.; Liu, J.; Zhang, X.; Wu, J.; and Cui, P. 2024. A Survey on Evaluation of Out-of-Distribution Generalization. *CoRR*.

Yue, X.; Zheng, T.; Zhang, G.; and Chen, W. 2024. MAmmoTH2: Scaling Instructions from the Web. *Advances in NIPS*.

Zhang, H.; Song, H.; Li, S.; Zhou, M.; and Song, D. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3): 1–37.

Zhang, X.; Li, J.; Chu, W.; Xu, R.; Yang, Y.; Guan, S.; Xu, J.; Jing, L.; Cui, P.; et al. 2025. On the Out-Of-Distribution Generalization of Large Multimodal Models. In *CVPR 2025*, 10315–10326.

Zhao, H.; Du, L.; Ju, Y.; Wu, C.; and Pan, T. 2025. Beyond IID: Optimizing Instruction Finetuning from the Perspective of Instruction Interaction and Dependency. In *Proceedings of the AAAI*, volume 39, 26031–26038.

Zhao, W.; Ren, X.; Hessel, J.; Cardie, C.; Choi, Y.; and Deng, Y. 2024a. WildChat: 1M ChatGPT Interaction Logs in the Wild. In *The Twelfth International Conference on Learning Representations*.

Zhao, Y.; Du, L.; Ding, X.; Xiong, K.; Sun, Z.; Jun, S.; Liu, T.; and Qin, B. 2024b. Deciphering the Impact of Pretraining Data on Large Language Models through Machine Unlearning. In *Findings of the ACL 2024*, 9386–9406.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Li, T.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Li, Z.; Lin, Z.; Xing, E. P.; et al. 2023a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in NIPS*, 36: 46595–46623.

Zhong, Y.; Zhou, W.; and Wang, Z. 2025. A Survey of Data Augmentation in Domain Generalization. *Neural Processing Letters*, 57(2): 34.

Zhou, C.; Liu, P.; Xu, P.; Iyer, S.; Sun, J.; Mao, Y.; Ma, X.; Efrat, A.; Yu, P.; Yu, L.; Zhang, S.; Ghosh, G.; Lewis, M.; Zettlemoyer, L.; and Levy, O. 2023. LIMA: Less Is More for Alignment. *arXiv preprint arXiv:2305.11206*.

Zhou, C.; Wang, Z.; Du, B.; and Luo, Y. 2024. Cycle Self-Refinement for Multi-Source Domain Adaptation. In *Proceedings of the AAAI*, volume 38, 17096–17104.