FANNO: Augmenting High-Quality Instruction Data with Open-Sourced LLMs Only

Anonymous ACL submission

Abstract

Instruction tuning stands as a crucial advancement in leveraging large language models (LLMs) for enhanced task performance. However, the annotation of instruction datasets has traditionally been expensive and laborious, often relying on manual annotations or costly proprietary LLMs. Recent works explore approaches to synthesize data with open-sourced LLMs but require high-quality human-crafted seed data. In this work, we introduce FANNO, an end-to-end framework to synthesize high-quality instruction data with open-sourced LLMs and sampled unlabeled documents, eliminating the necessity for seed data. Starting from diverse pre-screened documents, the framework synthesizes complex and diverse high-quality instruction and response pairs in different stages. We propose a tagging-based prompt method to generate diverse and complex seed data and a UCB-based approach to augment more instruction data with the seed data. A novel *Think Different* prompt is proposed to address the distributional limitations of the seeds, further boosting the data diversity. Experiments on Open LLM Leaderboard, MT-bench, and AlpacaEval benchmarks prove that the FANNO can generate diverse and complex high-quality data even with a Mistral-7B-instruct as the teacher model. The synthesized instruction data demonstrates performance that is comparable to, or even surpasses, baseline annotation methods with proprietary LLMs or open-sourced LLMs while requiring fewer instruction data samples.

1 Introduction

Large language models (LLMs) have made significant contributions across numerous fields like math reasoning and open-end question answering (Zheng et al., 2024a; Wang et al., 2024a; Wettig et al., 2024; Fan et al., 2023). Instruction tuning (Ouyang et al., 2022) enhances the model's abilities to follow user instructions even in unseen tasks (Li et al., 2023b; Bai et al., 2023). The instruction data is expected to have high quality in terms of correctness, complexity, and diversity (Zhou et al., 2023; Liu et al., 2023a). However, the human-annotated instruction data is prohibitively expensive when human experts are incorporated (Ouyang et al., 2022; Chiang et al., 2023) or the data has suboptimal quality when collected via crowd-sourcing (Srivastava et al., 2022; Conover et al., 2023). Previous studies explore approaches to synthesize high-quality instruction data with either proprietary LLMs (Wang et al., 2022; Xu et al., 2023) or open-sourced LLMs (Li et al., 2024; Lou et al., 2024). While effective, existing methods often require access to expensive APIs or hand-crafted seed datasets, imposing significant financial and operational barriers to widespread use. When synthesized with open-sourced LLMs (Zheng et al., 2024c; Yehudai et al., 2024; Press et al., 2023), the data quality is less satisfactory as demonstrated by the benchmark performance of finetuned LLMs (Zheng et al., 2024b; Lin et al., 2024; Dubois et al., 2024; Wang et al., 2024b). These LLMs have a limited ability to follow instructions and provide insufficient control over the complexity and diversity of the synthesized data. For example, simply prompting open-sourced LLMs to generate diverse instructions (Wang et al., 2022; Xu et al., 2023) still produces data that closely resembles the original seed data.

In this work, we develop FANNO (Free ANNOtator), an end-to-end instruction data synthesis framework with sampled unlabeled documents and open-sourced LLMs only that enhances control over data quality. This framework methodically breaks down the annotation process into three distinct stages: document pre-screening, instruction generation, and response generation. The pre-screening stage enhances the diversity of raw documents in both topic and structure, acting as an effective probe for eliciting high-quality data from LLMs. The instruction generation stage generates high-quality

instructions through two steps: seed instruction generation and instruction augmentation. In the seed instruction generation phase, a tagging-based prompt is used to generate candidate seeds across various task types and difficulty levels, while in the instruction augmentation phase, the UCB algorithm autonomously selects high-quality instructions while maintaining an exploratory component for newly generated instructions. The proposed *Think Different* strategy further overcomes the distributional limitations of the initial seeds, enhancing the data diversity. Empirical evidence on Open LLM Leaderboard, AlpacaEval, and MT-bench benchmarks confirm the efficacy of FANNO framework on two 7B LLMs. The generated dataset is virtually indistinguishable from instruction datasets like **Alpaca-GPT4**, marking a significant stride in instruction data development.[1]

2 Related Work

Instruction Data Generation Two main approaches have been explored for instruction data creation: (1) **Human Annotation**, which leverages human expertise to design prompts and collect multi-task datasets spanning various categories (Srivastava et al., 2022; Conover et al., 2023). While producing high-quality data, manual annotation is effort-intensive and costly, especially for devising complex textual instructions. (2) **LLM-based Synthetic Data Generation** Recent research increasingly favors harnessing the creative capabilities of LLMs, such as GPT-4 (OpenAI, 2023), over human input for creating instruction-following datasets (Geng et al., 2023; Chiang et al., 2023). ALPACA (Taori et al., 2023) and ALPACA-GPT4 (Peng et al., 2023) have also used more powerful LLMs to enhance data quality. Another line of research involves generating task instructions from "seeds" and filtering (Wu et al., 2023). For example, WIZARDLM (Xu et al., 2023) employed an instruction evolution paradigm to increase seed instruction complexity and diversity, while SELF-INSTRUCT (Wang et al., 2022) used human-annotated instructions as demonstrations to guide LLMs in the instruction evolution process. OSS-instruct (Wei et al., 2024) generates diverse instruction data for code by utilizing open-source code snippets. LLM2LLM (Lee et al., 2024) selects difficult data for labeling with a teacher model. Genie (Yehudai et al., 2024) uses four manually labeled 'doc, instruction, response' pairs as in-context learning examples to generate new QA pairs. Mammoth2 (Yue et al., 2024) proposes to extract question-answer pairs already existing in the web-crawled data. Humpback (Li et al., 2024) generates instructions using vast amounts of unlabeled web text. These datasets are costly in terms of labor or proprietary model expenses. In contrast, FANNO maintains high instructional quality autonomously, utilizing open-source models efficiently with just a 7B model size.

Data Quality Enhancement Related works in the field of enhancing data quality have focused on several key aspects, such as instruction difficulty, diversity, and correctness. HUMPBACK (Li et al., 2024) and KUN (Zheng et al., 2024c) utilize the language model's capability in combination with tailored prompts for data filtering. In Addition, initiatives like GENIE (Yehudai et al., 2024) and MODS (Du et al., 2023) utilize specialized open-source LLMs for data filtering tasks. DEITA (Liu et al., 2023b) utilize fine-tuned large models to score the data for quality assessment. Moreover, efforts like ORCA-MATH (Mitra et al., 2024) and REFLECTION-TUNING (Li et al., 2023a) employ collaborative approaches with multiple LMs and self-reflection to enhance data quality.

3 Method

3.1 Overview of FANNO Framework

The FANNO framework aims to annotate correct, complex, and diverse instruction data with open-sourced LLMs given the sampled unlabeled document text. As depicted in Figure 1, FANNO consists of three pivotal stages: document pre-screening, instruction generation, and response generation. Unlike previous works (Wei et al., 2024; Yehudai et al., 2024) that generate the instruction and response at the same time within a single turn, FANNO propose to output them separately in different stages for better quality (see more discussion in Section 4.2).

Stage 1: Document Pre-Screening The unlabeled documents are sampled from a web-scale corpus or in-house text data. The documents are first segmented and deduplicated before being filtered based on quality, complexity, and diversity. The LLM-based filter is applied to address ambiguous content, privacy concerns, and advertisements (see Appendix D.1), thus improving the data quality. The length-based filter selects complex documents

[1] Our code, data, and model will be made public.

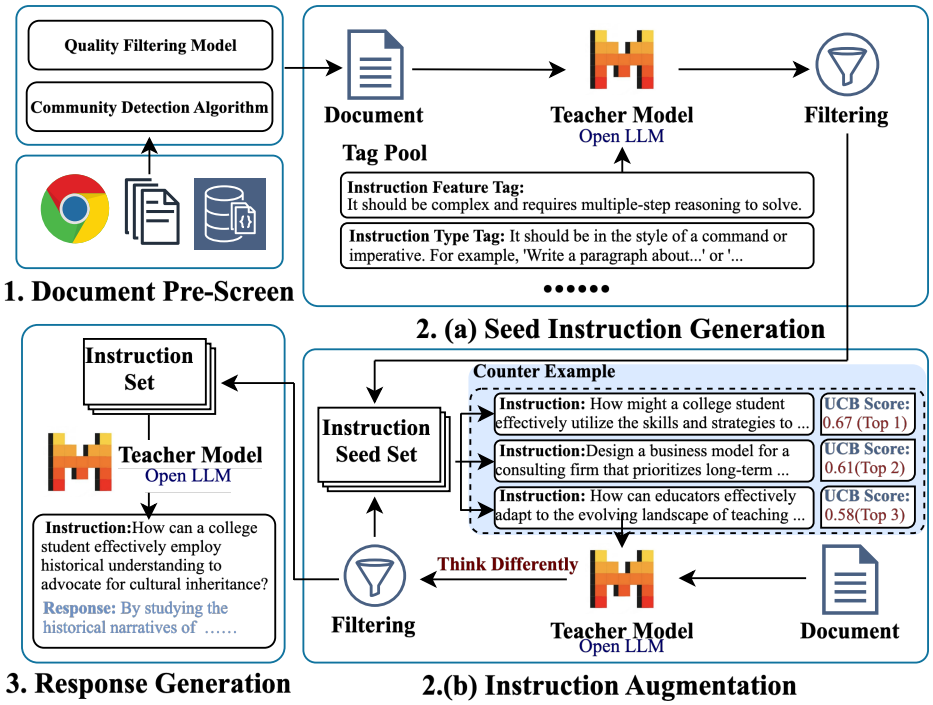

Figure 1: Overview of FANNO framework. **(1) Document Pre-Screening**: We process the unlabeled text data with filters and community detection algorithm. **(2a) Seed Instruction Generation**: FANNO generates seed instructions from pre-screened documents with diverse task types and difficulty levels through a tag pool. **(2b) Instruction Augmentation**: New instructions are augmented conditioned on the documents and few-shot examples selected from the seed instructions with the UCB algorithm. **(3) Response Generation**: The responses to instructions are generated directly by the teacher LLM.

inspired by the observation that text length correlates well with the complexity (Shen, 2024; Dubois et al., 2024; Liu et al., 2023a). The fast community detection algorithm (see Algorithm 1 in the appendix) further enhances the diversity of data by clustering the documents according to document-level embeddings (Reimers and Gurevych, 2019) and maintaining the representatives. More details are provided in Appendix B.2.

Stage 2: Instruction Generation The instructions are generated based on the pre-screened documents with the teacher model. To remove the requirements for human-crafted seed instruction data, FANNO proposes two distinct phrases for instruction generations: **Stage 2(a): seed instruction generation** and **Stage 2(b): instruction augmentation**. The seed instructions are generated with the tagging-based method (see Section 3.3) to enhance the data complexity and diversity. Then more diverse instruction data are augmented with the iterative UCB (Robbins and Monro, 1951, Upper Confidence Bound) method (see Section 3.2). Both generated instructions are filtered with the LLM-based method (see Table 5 in the appendix) to ensure the data quality. Appendix B.3 shows the detailed process.

Stage 3: Response Generation Different from previous work (Li et al., 2024), which directly applies the sampled document as a response to the generated instruction, FANNO directly prompts the teacher LLM to generate the response to each generated instruction. Inspired by the assumption that instruction tuning aims to activate the instruction-following abilities of LLMs instead of learning new knowledge (Zhou et al., 2023), we sample the documents from open-sourced pretraining dataset. Thus the teacher LLM is expected to answer the synthesized questions directly with its inherent knowledge.

The key challenge to instruction data synthesis is to improve the correctness, complexity, and diversity of the generated data (Zhou et al., 2023; Liu et al., 2023a). In the FANNO framework, the instruction is generated conditioned on the sampled document, which alleviates the hallucination problem and improves the correctness. The seed instructions are created with the requirements of different complexity levels, and more instructions are en-

riched with complex in-context examples selected with the UCB method (explained below). These methods improve the complexity of the generated data. Next, we will detail strategies to enhance the data diversity with the UCB-based method (Section 3.2) and the prompt-based method (Section 3.3).

3.2 UCB-Based Diversification Strategy

During the stage of instruction augmentation, the FANNO framework iteratively applies **UCB** (Robbins and Monro, 1951, Upper Confidence Bound) method to score and select high-quality instructions as in-context examples instead of randomly sampling (Wang et al., 2022). The instruction length is selected as the approximate metric of data quality as suggested by previous works (Shen, 2024; Dubois et al., 2024; Liu et al., 2023a). Given a seed instruction x_s, its UCB score is calculated as $UCB(x_s) = \bar{x}_s + C\sqrt{\frac{2\ln N}{n_s}}$, where $\bar{x}_s$ is the seed's average quality which is represented by the token length, N is the total iterations (i.e. number of samples synthesized so far), and C is a constant. The score encourages the selection of high-quality seed instructions that are less frequently chosen. The constant C is used to balance exploration and exploitation. In each iteration, we select k seeds with the highest UCB scores. The detailed process can be found in Appendix B.3. Ablation studies show that the UCB bootstrap method generates higher-quality data compared to random sampling, as demonstrated in Figure 4.

3.3 Prompt-based Diversification Strategies

The FANNO framework proposes two prompt-based strategies to enhance data diversity, which are illustrated below.

Tagging-Based Diversity-Enhanced Method The FANNO generates seed instruction data based on the pre-screened documents, eliminating the requirement of human-crafted seed data. To produce a set of diverse and complex instructions as the initial seeds, FANNO designs the prompt from two perspectives to enhance the data diversity: **Task Types** and **Difficulty Levels**, for which we have manually created the corresponding tags (see Appendix D.4). For each document, we traverse all combinations of tags of task types and instruction difficulty levels to generate seed instructions by $p(s|d, t_{\text{ty}}, t_{\text{df}})$, where s is the generated seed instruction, d is the sampled document, $t_{\text{ty}}, t_{\text{df}}$ are the tag selected from different task types and difficulty levels, respectively. The tagging-based diversification method enhances the control over the diversity of generated data as it provides detailed corresponding requirements on its complexity and task type.

Prompt Strategy to Encourage Diverse Thinking Following common practices (Wang et al., 2022; Xu et al., 2023), FANNO uses in-context examples to augment more instructions from seed data. Unlike conventional methods that use the examples as *positive* examples, our strategy views these examples as **negative** examples and asks the teacher LLM to propose different instructions. As shown in Table 10, our *Think Different* strategy extends the conventional prompt by inserting **"Please think differently"** in the prompt, meanwhile transforming the label from "### Example" to "### Counterexample". The proposed method encourages the teacher LLM to generate more diverse instructions by regarding the in-context examples as counterexamples, thus eliminating the probability of similar generations. The effectiveness of this method is further discussed in Section 5.3.

4 Experiments

4.1 Experiment Setup

Unlabeled Data The web-scale FALCON REFINED WEB corpus[2] (Penedo et al., 2023) includes text data from various domains. The first 500k documents of the corpus are used as our unlabelled data.

Dataset Annotation Details We choose the Mistral-7B-instruct-v0.3 (Jiang et al., 2023) as the teacher model for data annotation in all experiments. The model paraphrase-MiniLM-L6-v2 (Reimers and Gurevych, 2019)) is used to extract text embedding for clustering and filtering. Our annotated dataset FANNO contains 10k high-quality instruction-response pairs. We use greedy decoding with temperature 0 to generate the annotation results. All the prompts used are detailed in Appendix D.

Baselines We compare the instruction data synthesized with FANNO framework with other instruction data which are annotated with proprietary LLMs, including Alpaca-52k (Taori et al., 2023),

[2] https://huggingface.co/datasets/tiiuae/falcon-refinedweb

Model	Data Size	Open LLM Leaderboard					MT-bench
		ARC	HellaSwag	MMLU	TruthfulQA	Average	
Finetuning LLaMA-2-7b-base							
LLaMA-2-7B-Chat	–	54.10	78.65	45.69	44.59	55.76	6.27
Alpaca-52k	52k	54.78	78.17	46.65	55.26	45.03	4.14
Alpaca-Cleaned	51.8k	56.40	80.16	47.32	50.53	58.60	4.46
Alpaca-GPT4	52k	56.66	78.78	46.96	51.02	58.36	4.32
LIMA	1k	53.50	80.45	42.29	45.12	55.34	4.80
WizardLM-70k	65k	54.01	78.66	45.61	38.99	54.32	5.55
Muffin	68k	48.81	75.42	40.61	42.62	51.87	4.44
OSS-Instruct	20k	54.44	78.62	44.69	46.02	55.94	4.41
Genie	20k	55.55	78.62	44.68	40.77	54.91	4.24
Humback M_0*	–	56.31	81.20	47.45	47.59	58.14	5.07
Humback M_1*	–	52.99	78.57	45.48	41.45	54.65	–
FANNO	10k	54.86	77.62	46.85	53.71	58.26	5.64
Finetuning LLaMA-3-8b-base							
Alpaca-Cleaned	51.8K	56.83	78.92	57.63	50.49	60.97	5.67
LIMA	1k	56.49	82.11	63.27	46.96	62.21	5.74
WizardLM-70k	70k	53.75	78.83	58.09	46.53	59.30	5.71
Muffin	68k	47.61	74.43	36.81	43.82	50.67	4.91
OSS-Instruct	20k	56.16	78.64	57.91	51.74	61.11	5.54
Genie	20k	51.45	77.10	44.95	37.16	52.67	3.03
FANNO	10k	59.13	77.55	60.50	55.35	63.13	6.52

Table 1: Comparison results on Open LLM Leaderboard and MT-bench. By default, we report the Open LLM Leaderboard results with the lm-evaluation-harness toolkit. As Humback does not release its checkpoint, we compare it on the official Huggingface Open LLM Leaderboard platform and denote this setup as *. Data size represents the number of samples in the instruction data.

Alpaca-Cleaned[3], Alpaca-GPT4 (Peng et al., 2023), LIMA (Zhou et al., 2023), WizardLM-70k (Xu et al., 2023), and Muffin (Lou et al., 2024). We also compare FANNO with other annotation methods, including OSS-instruct (Wei et al., 2024), Genie (Yehudai et al., 2024), and Humpback (Chen et al., 2024). Please refer to Appendix A for more details about the baselines.

Base Model and Finetuning Setup We compare FANNO against baseline datasets on finetuning LLaMA-2-7b-base and LLaMA-3-8b-base model (Touvron et al., 2023). We perform supervised instruction tuning using a fully finetuning method except in the ablation study where the LoRA (Hu et al., 2021) method is applied. Please refer to Appendix C for detailed configurations.

Bechmarks We evaluate each finetuned model on the following three benchmarks.

• **Open LLM Leaderboard** (Beeching et al., 2023) includes ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), and TruthfulQA (Lin et al., 2022) task. We utilize the lm-evaluation-harness toolkit[4] (Gao et al., 2023) for evaluating different models to maintain consistency with the official setup. As Humpback does not release its checkpoint, we compare with its performance shown in the official Huggingface Open LLM Leaderboard.

• **MT-Bench** (Zheng et al., 2024b) is aimed at assessing the conversational and instruction-following abilities of LLMs with GPT-4 as the automated evaluator. A one-shot chat template is used to test all models in our experiments.

• **AlpacaEval 2.0 Benchmark** (Li et al., 2023b; Dubois et al., 2024) is an automated evaluation framework based on a annotation model(GPT-4). By comparing responses generated by two different models for the same set of 805 prompts, AlpacaEval computes the pairwise win rate, automating the evaluation process.

[3] https://huggingface.co/datasets/yahma/alpaca-cleaned

[4] https://github.com/EleutherAI/lm-evaluation-harness

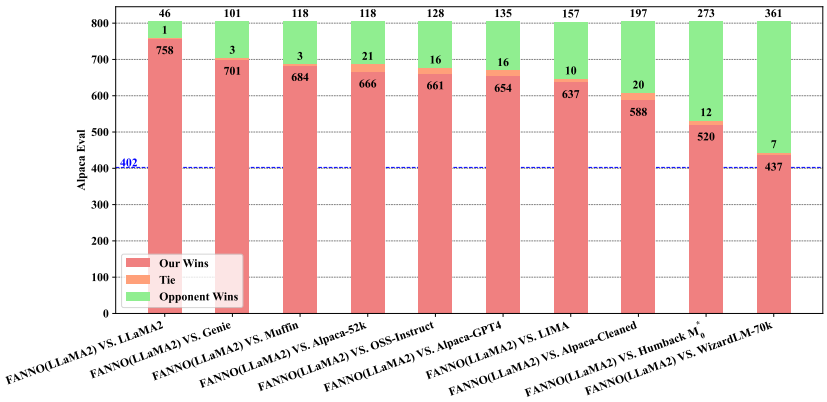

Figure 2: Results on Length-Control AlpacaEval2.0 Benchmark

4.2 Main Results

We presents the experiment results in Table 1 and Figure 2. We include the results of the Official LLaMA-2-Chat in Table 1 for reference. As demonstrated, FANNO outperforms nearly all baselines fine-tuned with LLaMA-2-7B and LLaMA-3-8B base models across various benchmarks, highlighting its effectiveness.

Compared to the instruction data annotated with proprietary LLMs, FANNO outperforms all baselines on LLaMA-3-8B-base, exceeding LIMA by 0.92 points on the Open LLM Leaderboard and by 0.78 points on MT-bench. Additionally, it surpasses Alpaca-Cleaned by 2.16 points on the Open LLM Leaderboard and by 0.85 points on MT-bench. For LLaMA-2-7B-base, FANNO outperforms all baselines across all benchmarks, except that it achieves comparable scores with Alpaca-Cleaned and Alpaca-GPT4 on the Open LLM Leaderboard. However, these two datasets are five times larger than FANNO.

Compared to baseline data annotation methods with open-sourced LLMs, FANNO consistently outperforms all baselines across all benchmarks. For instance, it exceeds the best baseline, Humback M_0, by 0.57 points on MT-bench when fine-tuning LLaMA-2-7B-base. Moreover, FANNO is much simpler than baseline annotation methods, as it does not require iterative optimization of the teacher LLM, unlike Humback. Compared to OSS-instruct and Genie, which directly generate instructions and corresponding responses from raw documents or code snippets within a single turn, FANNO enhances the control process in data generation by separate instruction and response generation, resulting in significantly better outcomes. The annotated FANNO instruction dataset is significantly smaller than all baselines except LIMA, indicating high data efficiency, as it achieves better performance with less data compared to baselines that utilize larger datasets. Additionally, the data annotation process for FANNO is fully automated and end-to-end, unlike LIMA, which requires manual filtering of the dataset.

ID	Configuration	Open LLM Leaderboard ARC	HS	MMLU	TQA	Avg.
Pre-screening						
(0)	(1) w/o PSR	54.44	78.66	44.69	46.02	55.95
Instruction Ablation						
(1)	(7) w/o TG&TD&UCB	55.46	78.51	46.00	45.85	56.44
(2)	(7) w/o TG	55.46	78.45	45.03	50.12	57.27
(3)	(7) w/o TD	54.69	79.18	45.92	50.19	57.50
(4)	(7) w/o UCB	55.63	79.43	44.84	51.16	57.77
Response Ablation						
(5)	FANNO (OD)	55.46	78.31	44.99	45.68	56.11
(6)	FANNO (RAG)	55.03	78.46	47.02	46.26	56.69
(7)	FANNO	55.63	79.45	46.84	51.01	58.23

Table 2: Ablation results from the lm-evaluation-harness (Gao et al., 2023). (0) Basic framework: (1) without pre-screening (PSR); (1) FANNO without Tag (TG), Think Different (TD) and UCB-selection (UCB); (2) FANNO without Tag in seed instruction generation; (3) FANNO without Think Different; (4) FANNO without UCB-selection; (5) Response generation with the original document (OD); (6) Response generation with retrieved document (RAG); (7) The complete version of FANNO, for which the response are directly generated without document.

4.3 Ablation Studies

We conduct ablation studies to identify the key components of FANNO, including the pre-screening stage (PSR), the Tag in seed instruction generation (TG), and the Think Different (TD) and UCB-selection (UCB) strategies in instruction augmentation. As shown in Table 2, all components are essential for FANNO. Specifically, removing the

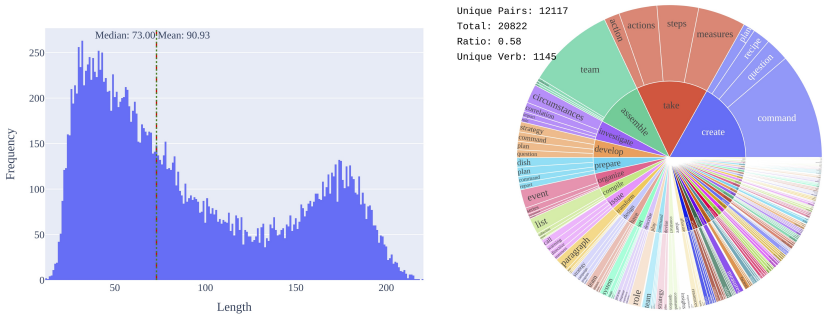

Figure 3: Left: Instruction length distribution of FANNO; Right: Top 50 common verbs and their corresponding nouns of FANNO. We compare FANNO with Alpaca-Cleaned, for which the detailed results are in Figure 6 and 8.

Tag, Think Different, and UCB-selection strategies results in performance declines of 0.96, 0.73, and 0.46 points, respectively, while removing all of these components leads to a total performance decline of 1.79 points. Additionally, further removing the pre-screening stage results in a performance decline of 0.49 points.

We also conduct an ablation study on the response generation stage to explore whether documents are necessary for generating responses. We experiment with three variants: using the original document (OD), the retrieved document (RAG), and no document at all during the response generation stage. The results indicate that using either the original or retrieved document leads to significant performance declines. As discussed in Section 3.1, the user queries are expected to be directly answered by the teacher LLM. However, the extra information provided by the original document or retrieved document might contain interference and biased information that leads to different responses. The conflict between the inherent LLM knowledge and external document knowledge might harm the performance. Since the annotated data is used for instruction tuning where no documents are provided, it is better to exclude the document information during the response generation process.

5 Analyses

5.1 Data Statistics

We provide the data statistics of FANNO from the perspectives of length, diversity, quality, and complexity. We also report the statistics of the best-performing baseline dataset, Alpaca-Cleaned, for comparison.

Length To study the distribution of the length of instructions, we tokenize each instruction combined with input and count the words within it as its length. Figure 3 left illustrates the distribution of instruction length for FANNO, while that of Alpaca-Cleaned is in Figure 6 of Appendix E.1. The results show that FANNO instructions are more balanced than Alpaca-Cleaned, and the mean value of lengths is higher than that of Alpaca.

Diversity We conduct the task diversity analysis using the approach from SELF-INSTRUCT (Wang et al., 2022). Figure 3 shows the distribution of the verb-noun structure of instructions for FANNO, while that of Alpaca-Cleaned is in Figure 8. Overall, FANNO has a higher proportion of unique verb-noun pairs (0.58 vs. 0.31) compared to Alpaca-Cleaned. The verb-noun pairs in FANNO (e.g., assemble-team, create-command, take-measures) are also more challenging than those in Alpaca-Cleaned (e.g., rewrite-sentence, generate-list, create-list). This suggests that FANNO is a more diverse and complex dataset than Alpaca-Cleaned.

Quality and Complexity To evaluate the quality and complexity of instruction-response pairs, we utilize Deita-quality-scorer model and Deita-complexity-scorer model (Liu et al., 2023b) as an evaluator to score our instructions. Figure 9 in Appendix E.1 shows the quality and complexity comparison between FANNO and Alpaca-Cleaned, of which the result shows that FANNO instructions possess a more balanced complexity distribution and higher average quality.

5.2 UCB-Based Strategy Improve Instruction Complexity and Diversity

UCB Bootstrap is employed to stabilize the process of instruction improvement. We monitor the diversity and complexity of instructions across different

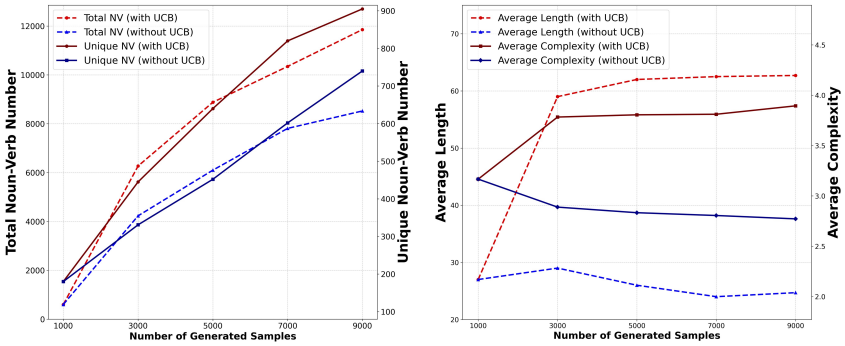

Figure 4: Diversity (left) and complexity (right) of the annotated datasets as iterations increase.

iterations and compare it with a random selection strategy. For diversity, we measure it with the number of noun-verb pairs following Wang et al. (2022). For complexity, we measure it with averaged instruction length, as well as the complexity score using the Deita-complexity-scorer model in Liu et al. (2023b).

As depicted in Figure 4, we observe a shaper increase in both the average complexity and diversity scores as the iteration progresses compared with the ablation method (without UCB, namely random sampling), demonstrating the effectiveness of UCB-based strategy. By prioritizing the exploration of longer and newly generated instructions for few-shot examples of instruction generation, UCB facilitates the annotation of more challenging and creative instructions.

5.3 *Think Different* Improves Instruction Diversity

To gain a comprehensive understanding of the role of *Think Different* in promoting diversity, we evaluate its impact using four metrics: (1) Number of unique noun-verb pairs, (2) Number of unique intent tags following Lu et al. (2023), (3) Averaged instruction length and (4) The Length Distribution Index (LDI). LDI measures the length distribution shift between the generated instruction dataset and the original seed dataset, calculated via KL divergence. We begin with 175 instruction entries sourced from the self-instruct seed set [5] and extend this to a collection of 5k instruction samples using different approaches.

As shown in Table 3, *Think Different* effectively annotates a new instruction dataset that is more diverse than the seed dataset. It also surpasses baseline annotation methods such as Self-Instruct and WizardLM in enhancing data diversity. Additionally, it outperforms the two variants of Think Different: one that removes the '### Counter Example' (CE) and the other that omits the 'Please Think Differently' tag (PTD) in the prompt. This underscores the significance of these components in enhancing the teacher model's ability to synthesize high-quality, diverse instruction data.

ID	Method	NV pairs	Intent Tags	Avg Len	LDI
(1)	*Seed*	143	248	37.45	–
(2)	*Think Different*	755	**2141**	**138.85**	**17.21**
(3)	*(2) – 'CE'*	53	859	84.32	13.03
(4)	(2) – 'PTD'	55	779	47.96	12.21
(5)	Self-instruct	60	916	83.01	13.43
(6)	wizardlm	**997**	1299	135.47	15.54

Table 3: Diversity comparison of instruction generation methods. 'CE' and 'PTD' denote the strategy of transforming the label from "### Example" to "### Counterexample" and prepending "Please think differently" in the prompt, respectively.

6 Conclusion

The development of instruction data has been hindered by the high cost and labor-intensive nature. In this paper, we introduced FANNO, an autonomous and low-cost end-to-end framework that addresses these challenges by streamlining the annotation process with open-sourced LLMs. FANNO efficiently generates datasets of high quality, diversity, and complexity through a structured process involving pre-screening, instruction generation, and response generation. This unified process eliminates the need for pre-existing annotated data or costly API calls, advancing the instruction data development. Empirical experiments validate the efficacy of FANNO, underscoring the framework's potential to democratize access to high-quality instruction datasets.

[5] https://github.com/yizhongw/self-instruct

Limitations

While FANNO has demonstrated outstanding performance, several limitations must be acknowledged. The responses are not entirely dependent on the document, leading to the introduction of certain hallucinations in the fine-tuning data. This suggests that the model's reliance on the provided context needs to be strengthened to improve factual consistency. The simplistic approach of equating instruction length with its value is rather crude. The true value of an instruction is influenced by various factors such as difficulty, quality, and novelty. Future work will aim to develop a more nuanced understanding and evaluation of instruction value. The quality of generated instructions is contingent upon the capabilities of both the generator and the evaluator. This process is sensitive to the teacher model and the prompts used, indicating a need for designing prompts that are specifically tailored to the model. Addressing these limitations will be a focus of our future work.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Mike Conover, Matt Hayes, Matt Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Ali Ghodsi, Patrick Wendell, and Patrick Zaharia. 2023. Hello dolly: Democratizing the magic of chatgpt with open models.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. 2023. Mods: Model-oriented data selection for instruction tuning. *Preprint*, arXiv:2311.15653.

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

Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large language models for software engineering: Survey and open problems. *Preprint*, arXiv:2310.03533.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

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

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. Llm2llm: Boosting llms with novel iterative data enhancement. *Preprint*, arXiv:2403.15042.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Heng Huang, Jiuxiang Gu, and Tianyi Zhou. 2023a. Reflection-tuning: Data recycling improves llm instruction-tuning. *ArXiv*, abs/2310.11716.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. 2024. Self-alignment with instruction backtranslation. *Preprint*, arXiv:2308.06259.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *Preprint*, arXiv:2406.04770.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

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

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

Renze Lou, Kai Zhang, Jian Xie, Yuxuan Sun, Janice Ahn, Hanzi Xu, Yu Su, and Wenpeng Yin. 2024. MUFFIN: Curating multi-faceted instructions for improving instruction following. In *The Twelfth International Conference on Learning Representations*.

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. *Preprint*, arXiv:2308.07074.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *Preprint*, arXiv:2402.14830.

OpenAI. 2023. 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, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *Preprint*, arXiv:2306.01116.

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

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. *Preprint*, arXiv:2210.03350.

Sebastian Raschka. 2023. Finetuning llms with lora and qlora: Insights from hundreds of experiments. *Lightning AI*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Ming Shen. 2024. Rethinking data selection for supervised fine-tuning. *Preprint*, arXiv:2402.06094.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. *arXiv preprint arXiv:2206.04615*.

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

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

Siyuan Wang, Zhongyu Wei, Yejin Choi, and Xiang Ren. 2024a. Can llms reason with rules? logic scaffolding for stress-testing and improving llms. *Preprint*, arXiv:2402.11442.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, and Others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Preprint*, arXiv:2406.01574.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. *Preprint*, arXiv:2312.02120.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. Qurating: Selecting high-quality data for training language models. *Preprint*, arXiv:2402.09739.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2023. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *arXiv preprint arXiv:2304.14402*.

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*.

Asaf Yehudai, Boaz Carmeli, Yosi Mass, Ofir Arviv, Nathaniel Mills, Assaf Toledo, Eyal Shnarch, and Leshem Choshen. 2024. Genie: Achieving human parity in content-grounded datasets generation. *Preprint*, arXiv:2401.14367.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *Preprint*, arXiv:2405.03548.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024a. Take a step back: Evoking reasoning via abstraction in large language models. *Preprint*, arXiv:2310.06117.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Tianyu Zheng, Shuyue Guo, Xingwei Qu, Jiawei Guo, Weixu Zhang, Xinrun Du, Chenghua Lin, Wenhao Huang, Wenhu Chen, Jie Fu, et al. 2024c. Kun: Answer polishment for chinese self-alignment with instruction back-translation. *arXiv preprint arXiv:2401.06477*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

A Experiment Baselines

• Alpaca-52k (Taori et al., 2023). This dataset is developed by Stanford University using Text-Davinci-003. It encompasses 52,002 instruction-following samples.

• Alpaca-GPT4 (Peng et al., 2023). This dataset contains English Instruction-Following Data generated by GPT-4 using Alpaca prompts for fine-tuning LLMs. It encompasses 52,002 instruction-following samples, the same as Alpaca-52k.

• Alpaca-Cleaned. This is a cleaned version of the Alpaca-GPT4 Dataset to address problems like hallucinations, merged instruction, and so on. It encompasses 51,760 instruction-following samples.

• LIMA (Zhou et al., 2023).This is a dataset of 1,000 prompts and responses from a variety of sources, primarily split into community Q&A forums and manually authored examples, where the outputs (responses) are stylistically aligned with each other, but the inputs (prompts) are diverse.

• WizardLM-70k (Xu et al., 2023). This dataset employs the Evol-Instruct algorithm to enhance the quality of instruction data. Incorporating ChatGPT during the reformulation phase ensures data fidelity. Among its 250,000 instructions, we primarily focused on the WizardLM-7b subset, which consists of 70,000 samples.

• Muffin (Lou et al., 2024) MUFFIN's data curation includes input sampling, instruction collection via two methods, output annotation by ChatGPT/GP4-4, instruction filtering, and classification expansion. This is a large dataset of 68k training instances.

• Humpback. This self-alignment method generates instruction data through reverse fine-tuning.

• Oss-instruct. A method leveraging open-source code snippets to generate diverse instruction data, aiming to mitigate the bias in synthetic data by utilizing open-source references for more realistic outputs.

• Genie. A method for automatically generating content-grounded data involving content preparation, task-specific generation (e.g., Q&A or summaries), and a filtering mechanism to ensure quality and faithfulness.

B FANNO Details

B.1 Pre-screen Details

Our objective was to efficiently enhance the selection process, minimizing time spent while maximizing quality outcomes. Initially, we employed **Mistral-7b-instruct-v0.3** (Jiang et al., 2023) to evaluate texts for repetitive content, personal privacy concerns, specific themes, and advertising, using prompts to guide scoring and annotation (see Table 4). For diversity assessment, we utilized a fast community detection algorithm 1 with hyperparameters set to k = 2 and simratio = 0.7(k: the minimum size of a community; simratio: controls the similarity threshold, Only node pairs with similarity scores higher than this threshold are considered connected), facilitating the classification of half a million entries within minutes. The model paraphrase-MiniLM-L6-v2 (Reimers and Gurevych, 2019)) is used for text embedding. For larger datasets, texts were segmented into groups for individual community detection analyses. After the pre-screening process, *Pre-Screen Data* has approximately 30k records, which is 6% of the original. This stage was designed to balance the trade-off between processing speed and analytical precision, prioritizing efficiency over exhaustive detail examination.

B.2 Fast Community Detection Algorithm

As Algorithm 1 has shown, the Fast Community Detection Algorithm is used to cluster the embeddings of instructions processed by SentenceTransformer (Reimers and Gurevych, 2019), which can then represent the diversity of instructions. Specifically, Fast Community Detection works by iteratively identifying groups of data points (embeddings of sentences) that are closely related based on a predefined similarity threshold, efficiently leveraging cosine similarity calculations. It prioritizes larger communities while minimizing overlapping clusters to produce meaningful community structures.

Algorithm 1 Fast Community Detection (Reimers and Gurevych, 2019)

```
function COMMUNITYDETECTION(embeddings, threshold, min_community_size, batch_size)
    Normalize embeddings
    Initialize extracted_communities as empty list
    for start_idx in range(0, length(embeddings), batch_size) do
        Compute cosine similarity scores for batch starting from start_idx
        Find top-k values from cosine similarity scores
        for i in range(length(top_k_values)) do
            if last element of i-th top-k values ≥ threshold then
                Find top-k most similar entries for i-th element
                while last element of top-k values > threshold and sort_max_size < length of embeddings do
                    Increase sort_max_size if needed
                end while
                Add indices of entries with similarity ≥ threshold to extracted_communities
            end if
        end for
    end for
    Sort extracted_communities by size
    Remove overlapping communities from extracted_communities
    return extracted_communities
end function
```

B.3 Detailed Process of Instruction Generation

We show a detailed process of instruction generation below. The setup comprises a language model G parameterized by θ_G for generating instructions, a critic model J parameterized by θ_J for evaluating instruction quality, as well as a document set $\mathcal{D}$, a subset $\mathcal{D}^{'}$, task-type tags $\mathcal{T}_{\mathcal{T}}\mathcal{Y}$, and difficulty-level tags $\mathcal{T}_{\mathcal{D}}\mathcal{F}$.

1. **Initialization:**

$$S \leftarrow \emptyset$$

2. **Seed Generation ($SeedGen$):**

$$\begin{aligned} &\forall d \in \mathcal{D}^{'}, generate\ s \sim P(s|d;\theta_G) = P(t)P(s|d,t;\theta_G) \\ &\quad where\ t \sim \mathcal{U}(\mathcal{T}_{\mathcal{T}}\mathcal{Y} \times \mathcal{T}_{\mathcal{D}}\mathcal{F}) \\ &\quad S \leftarrow S \cup \{s\} \end{aligned}$$

3. **Instruction Augmentation ($InsAug$):** For f rounds or until $|S|$ reaches a desired threshold:

   a. Select a subset $S' \subset S$ using the UCB strategy:

$$\begin{aligned} UCB(s) &= \bar{x}_s + C\sqrt{\frac{2\ln N}{n_s}} \\ S' &= \{s_i | s_i \in S, UCB(s_i) \text{ is maximized}\} \end{aligned}$$

   where $\bar{x}_s$ is the average quality score of instruction s, N is the total number of iterations, C is a hyper-parameter constant used to control exploration, n_s is the number of times instruction s has been selected.

b. Generate new instructions given sampled examples and filter out those that are similar to the examples:

$$x' \sim P(x|c, S'; \theta_G)$$
$$\text{s.t. } Sim(x'; s_i) < \tau \text{ for all } i$$

where τ is a similarity threshold.

c. Update S with the augmented instructions:

$$S \leftarrow S \cup \{x'\}$$

B.4 Ablation of Think Different Strategy

- **Noun-Verb Pairs**: Inspired by previous work (Wang et al., 2022), the diversity of an instruction dataset is represented by the total number of unique noun-verb pairs within the dataset. This serves as a key metric for assessing linguistic variety.
- **Instruction and Response Token Distribution**: This evaluates the lexical diversity within the instructions and their corresponding responses by analyzing the token distribution. A broader token distribution indicates more lexical variety.
- **Intent Tags**: Following the methodology from Instagger[6], the semantic diversity of generated instructions is the total number of intent tags. This ensures coverage across a wide range of task types and questioning styles, capturing a more diverse set of instructional intents.

C Experiment Setting Details

We use the same hyperparameters as existing supervised instruction tuning methods (Chiang et al., 2023; Raschka, 2023). Specifically, we use a cosine learning rate scheduling strategy with a starting learning rate of 2×10^{-5} and a weight decay of 0.1 to optimize the training process. The batch size per device is set to 128, and a gradient accumulation of 8 steps is applied, resulting in an effective batch size of 128. The model is trained for three epochs, utilizing full-shard data parallelism (FSDP) with auto-wrapping for LlamaDecoderLayer and bf16 precision enabled. Additionally, we apply a dropout rate of 0.1 for regularization purposes. The warmup ratio is set at 0.03 for stabilization during early training. For the LoRA configuration, we employ a rank of 256 and set α to 512, with an initial learning rate of 5×10^{-5}. We utilize 8 NVIDIA A800 GPUs to train our model.

D Prompt Templates Used in FANNO

D.1 Text Filtering

[6] https://huggingface.co/OFA-Sys/InsTagger

Table 4: Prompts for Pre-Screen

```
You are act as a assistant to check useless, informal or ambiguous information. Let's think step
by step.
The objective is to meticulously inspect the text to determine if it is useless, informal or
ambiguous text (e.g. random characters, ambiguous paragraph, broken sequence, informally organized
text, etc.)
Your response should be '1' (yes) if the text contains useless, informal or ambiguous information,
or '0' (no) if it does not, without providing any reasoning and explanation.

### Document:
{doc}

### Answer:
```

```
You are act as a assistant to check privacy information. Let's think step by step.
The objective is to meticulously inspect the text to determine if it contains any privacy
information (e.g. human names, phone numbers, addresses, etc.).
Your response should be '1' (yes) if the text contains privacy information, or '0' (no) if it does
not, without providing any reasoning and explanation.

### Text:
{doc}

### Answer:
```

```
I want you to act as an advertisement evaluator. Let's think step by step.
The objective is to meticulously inspect the text based on certain characteristics and decide
whether it is an advertisement or not.
Your response should be '1' (yes) if the text is an advertisement, or '0' (no) if it is not,
without providing any reasoning and explanation.

Evaluate the text considering these characteristics:

- Promotional language or sales pitch
- Mention of product or service benefits
- Call to action (e.g., "Buy now", "Subscribe")
- Pricing information or special offers
- Contact information or links for more details

<Answer Format>: 1 or 0

### Text:
{text}

### Answer:
```

Table 5: Prompts for instruction generation filter

```
I want you to act as an instruction evaluator. Please evaluate this instruction and respond with
'0' (bad) or '1' (good), without giving reasons.
Standard: A good instruction Must not involve recent or current events. Historical events are
fine.
Example1:
Instruction: Please analyze the recent COVID-19 outbreak.
Answer: 0 (Reason: recent)
Example2:
Instruction: What's happening in China in September 2023?
Answer: 0 (Reason: in September 2023)
Example3:
Instruction: Provide an account of events from last Monday night.
Answer: 0 (Reason: last Monday night)

### Instruction:
{instruction}
### Answer:
```

```
I want you to act as a instruction evaluator. Please evaluate this instruction and respond with
'0' (bad) or '1' (good), without giving reasons.
Standard: A good instruction must not include any private information like names, addresses, phone
numbers, etc, unless the person is historical or famous.
Example1:
Instruction: What is the name of the person who lives at 123 Main Street?
Answer: 0 (Reason: private information)
Example2:
Instruction: What is the name of the first president of the United States?
Answer: 1 (Reason: historical)
Example3:
Instruction: What is the address of the CEO of Microsoft?
Answer: 0 (Reason: private information)

### Instruction:
{instruction}
### Answer:
```

```
I want you to act as a instruction evaluator. Please evaluate this instruction and respond with
'0' (bad) or '1' (good), without giving reasons.
Standard: A good instruction is perfectly logical, and practical, and can be fully understood by a
human.
A bad instruction, likely generated by AI, is generally vague, weird, complex, and long. It may
seem to string unrelated words, topics, and tasks together.
Example1:
Instruction: Considering the health benefits of a non-dairy diet, how does the emotional response
of individuals vary when they attend social events where dairy-based foods are served?
Answer: 0
Example2:
Instruction: Create a multidisciplinary essay that explores the and historical origins of the
dish 'Shrimp Alfredo Pasta Bake'. Discuss the various ingredients, their origins. Additionally,
translate the recipe instructions from English to Spanish.
Answer: 0

### Instruction:
{instruction}
### Answer:
```

Table 4 shows the prompts for basic filtering, including filtering information with useless information, privacy information, or advertisement.

Table 5 shows the prompts for instruction generation filtering, including filtering instructions that are time-sensitive, asking for private information, or not answerable.

D.2 Complexity and Quality Scorer

Table 6: Prompt for quality scorer

```
You are a helpful assistant. Please identify the quality score of the Response corresponding to
the Question.
### Question:
{instruction}
### Response:
{output}
### Quality:
```

Table 7: Prompt for complexity scorer

```
You are a helpful assistant. Please identify the complexity score of the following user query.
### Query:
{instruction}
### Complexity:
```

As Table 6 and 7 have shown, the prompts are provided to deita-complexity-scorer and deita-quality-scorer model (Liu et al., 2023b).

D.3 Generating Instruction Pairs

FANNO employs 2 ways to generate instruction response:

- Question, Document to Answer: model infers answer with both question and related document.
- Question to Answer: The model infers the answer directly with the question, using its own knowledge.

Table 8: Question, Document to Answer

```
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while
being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous,
or illegal content. Please ensure that your responses are socially unbiased and positive in nature.
If a question does not make any sense, or is not factually coherent, explain why instead of
answering something not correct. If you don't know the answer to a question, please don't share
false information.
### Instruction: {question}.
### Paragraph: {doc}.
### Response:
```

D.4 Seed Generation

Listing 1: Seed Generation

```
def seed_gen(text):
```

Table 9: Question to Answer

```
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.
### QUESTION: {question}
### Response:
```

```
    reasoning_tag = "It should be complex and requires multiple-step reasoning to solve."
    critical_thinking_tag = "It demands critical thinking skills to analyze from various perspectives and evaluate multiple solutions."
    creativity_tag = "It necessitates creative thinking to devise innovative solutions beyond conventional approaches."
    interdisciplinary_tag = "It demands integrating knowledge from diverse disciplines to address its multifaceted nature."
    command_tag = "It should be in the style of a command or imperative. For example, 'Write a paragraph about...' or 'Describe the...'"
    question_tag = "It should be in the style of a question or interrogative. For example, 'What is the...?' or 'How do you...?'"

    nli_tag = "It is a Natural language inference question: Assessing if evidence supports a conclusion."
    commonsense_tag = "It is a Commonsense question: Predicting outcomes based on everyday knowledge."
    sentiment_tag = "It is a Sentiment analysis question: Determining emotional response to a given scenario."
    paraphrase_tag = "It is a Paraphrasing question: Rewording a statement while retaining its meaning."
    close_book_qa_tag = "It is a Close-book QA question: Answering factual queries using pre-existing knowledge."
    struc2text_tag = "It is a Structure to text question: Describing a process or concept in written form."
    summarization_tag = "It is a Summarization question: Condensing key information from a larger text."
    translate_tag = "It is a Translation question: Converting text from one language to another."
    implicit_reasoning_tag = "It is a Implicit reasoning question: Inferring reasons behind common behaviors."
    text_category_tag = "It is a Text categorization question: Identifying defining characteristics of a given text type."

    tags = [reasoning_tag, critical_thinking_tag, creativity_tag, interdisciplinary_tag]
    classify = [nli_tag, commonsense_tag, sentiment_tag, paraphrase_tag, close_book_qa_tag, struc2text_tag, summarization_tag, translate_tag, implicit_reasoning_tag, text_category_tag]
    types = [command_tag, question_tag]

    QUESTION_TEMPLATE = """You're proficient in crafting complex question. Generate only one question that adheres to the provided #Paragraph#.
    The question should meet the following criteria:
    0. The person answering the question cannot see the #Paragraph#[SYSTEM: IMPORTANT], so the question must not contain phrases like 'Given the information provided', 'Based on the provided information', or similar expressions that imply direct citations or references from #Paragraph#.
    1. {characteristic}.
    2. {type}.
    3. {classify}.

    ### Paragraph:
    {text}
```

```
    ### Question:
    """
    prompts = [QUESTION_TEMPLATE.format(characteristic=tag, type=type, text=text,
        classify=c) for tag in tags for c in classify for type in types]
    return prompts
```

Code 1 shows the process of generating seed with sampled tags, including task types and difficulty levels.

D.5 Think Different Prompt

Table 10: Prompt for Think Differently

```
You are a helpful assistant. Your task is to conceive a complex query inspired from the ### Paragraph.
Please think differently from the examples provided below in terms of expressions, question types,
and initial verbs.
### Counterexample:
<Example1>: {seed1}
<Example2>: {seed2}
<Example3>: {seed3}

### Paragraph:
{text}

### Question:
```

D.6 Self-Instruct Prompting Templates for Data Generation

Self-Instruct relies on the following prompting template in order to elicit the generation from language models.

```
Come up with a series of tasks:

Task 1: {instruction for existing task 1}
Task 2: {instruction for existing task 2}
Task 3: {instruction for existing task 3}
Task 4: {instruction for existing task 4}
Task 5: {instruction for existing task 5}
Task 6: {instruction for existing task 6}
Task 7: {instruction for existing task 7}
Task 8: {instruction for existing task 8}
Task 9:
```

Table 11: Prompt used for *Self-Instruct*

E Data Analysis

E.1 Quality, Length, and Diversity

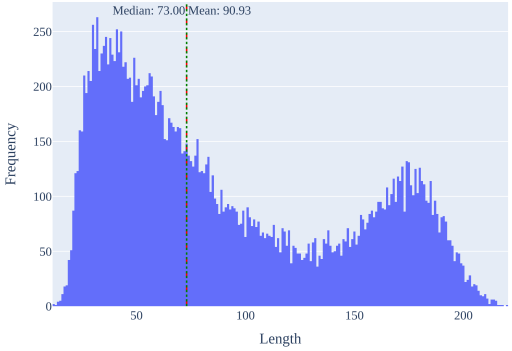

Figure 5: FANNO Instruction Length Distribution

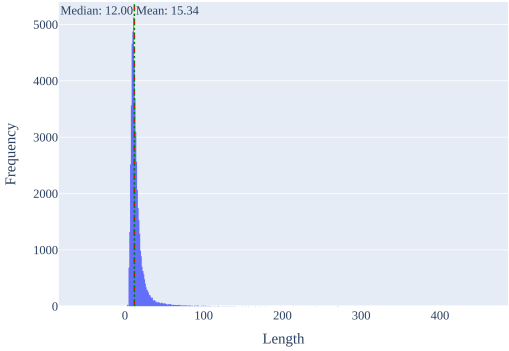

Figure 6: Alpaca-Cleaned Instruction Length Distribution

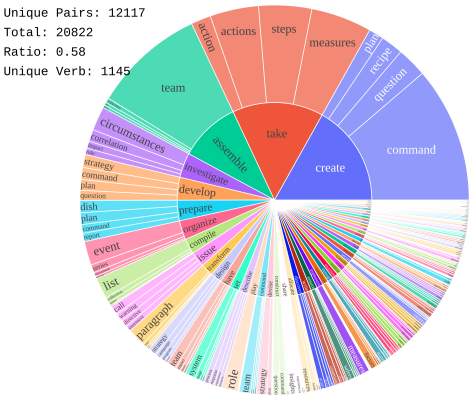

Figure 7: Top 50 common verbs and their corresponding nouns in FANNO

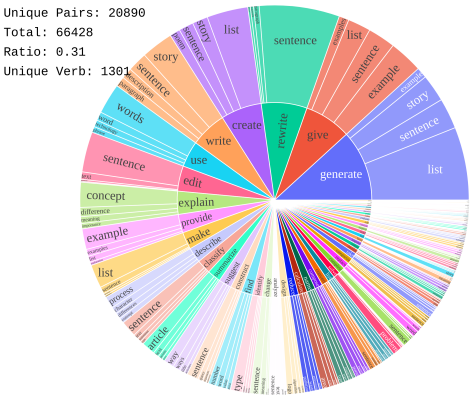

Figure 8: Top 50 common verbs and their corresponding nouns in Alpaca-Cleaned

Figures 5 and 6 show the word-level instruction length distribution of FANNO and Alpaca-Cleaned, respectively. Figures 7 and 8 show the verb-noun diversity of FANNO and Alpaca-Cleaned, respectively. Figure 9 shows the comparison of quality and complexity between FANNO and Alpaca-Cleaned.

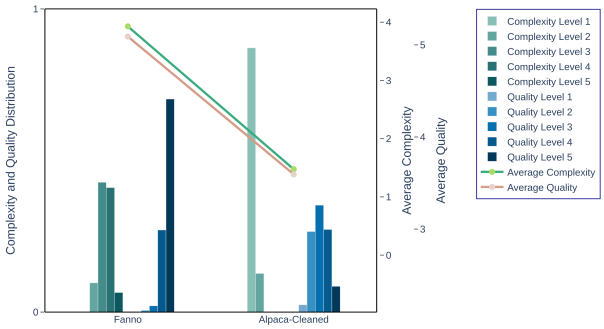

Figure 9: Quality and Complexity Comparison between FANNO and Alpaca-Cleaned