CYCLE-INSTRUCT: Fully Seed-Free Instruction Tuning via Dual Self-Training and Cycle Consistency

Zhanming Shen[♠◇], **Hao Chen**[♠◇], **Yulei Tang**[♣], **Shaolin Zhu**[♡]
Wentao Ye[♠], **Xiaomeng Hu**[♠], **Haobo Wang**[♠◇], **Gang Chen**[♠], **Junbo Zhao**[♠*]
[♠]Zhejiang University
[◇]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
[♣]University of Science and Technology of China
[♡]Tianjin University
{z.shen, j.zhao}@zju.edu.cn

Abstract

Instruction tuning is vital for aligning large language models (LLMs) with human intent, but current methods typically rely on costly human-annotated seed data or powerful external teacher models. While instruction back-translation techniques reduce this dependency, they remain fundamentally tethered to an initial seed set, which limits full automation, introduces biases, and can lead to inefficient use of unlabeled corpora. In this paper, we propose CYCLE-INSTRUCT, a novel framework that achieves fully seed-free instruction tuning. Inspired by cycle consistency, CYCLE-INSTRUCT employs a dual self-training loop where two models—an answer generator and a question generator—are bootstrapped solely from raw, unlabeled text. These models mutually supervise each other by reconstructing original text segments from their counterpart's generated pseudo-labels, effectively learning from the intrinsic structure of the data without any human-provided seeds. We demonstrate CYCLE-INSTRUCT's efficacy across four diverse data tracks, including general instruction-following, domain-specific tasks, dialogue logs, and plain text. Our extensive experiments show that CYCLE-INSTRUCT not only outperforms seed-driven back-translation baselines but also achieves performance comparable to strongly supervised methods.

1 Introduction

Instruction tuning (Ouyang et al., 2022) has emerged as a crucial technique for aligning large language models (LLMs) with human intent, enabling effective generalization across diverse instruction-based tasks (Ouyang et al., 2022; Wei et al., 2021; Touvron et al., 2023; Zhu et al., 2024a). However, conventional instruction tuning typically requires extensive human-annotated data (Köpf et al.; Conover et al., 2023) or relies on powerful external teacher models (Taori et al., 2023; Yin et al., 2023). These dependencies are not only costly and limit scalability (Wang et al., 2022) but are also often inapplicable in certain settings—for example, privacy-preserving scenarios (Zhang et al., 2024).

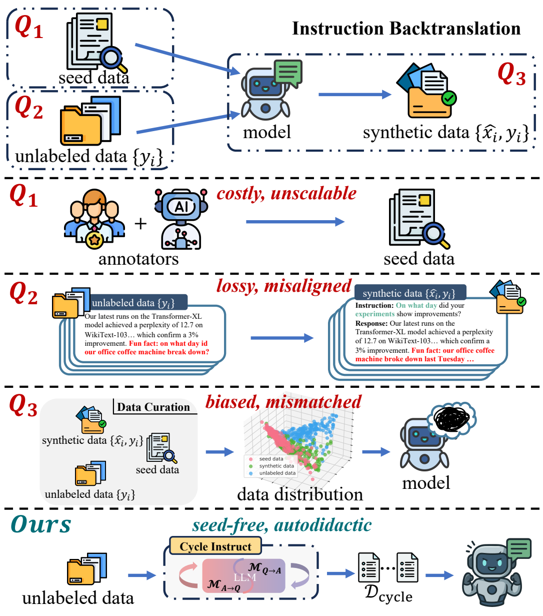

Q_1
seed data
Q_2
unlabeled data $\{y_i\}$
Instruction Backtranslation
model
Q_3
synthetic data $\{\hat{x}_i, y_i\}$
Q_1
costly, unscalable
annotators
AI
seed data
Q_2
lossy, misaligned
unlabeled data $\{y_i\}$
Our latest runs on the Transformer-XL model achieved a perplexity of 12.7 on WikiText-103… which confirm a 3% improvement. Fun fact: on what day id our office coffee machine break down?
synthetic data $\{\hat{x}_i, y_i\}$
Instruction: On what day did your experiments show improvements?
Response: Our latest runs on the Transformer-XL model achieved a perplexity of 12.7 on WikiText-103… which confirm a 3% improvement. Fun fact: our office coffee machine broke down last Tuesday …
Q_3
Data Curation
synthetic data $\{\hat{x}_i, y_i\}$
seed data
unlabeled data $\{y_i\}$
biased, mismatched
seed data
synthetic data
unlabeled data
data distribution
model
Ours
seed-free, autodidactic
Cycle Instruct
$\mathcal{M}_{Q\to A}$
LLM
$\mathcal{M}_{A\to Q}$
unlabeled data
$\mathcal{D}_{\text{cycle}}$

Figure 1: **Seed-dependency bottleneck:** (1) *Costly seed curation;* (2) *Data wastage & mis-paired;* (3) *Bias transfer & low diversity.*

Recent advances in *instruction back-translation* (Li et al., 2023a; Köksal et al., 2023) have sought to mitigate above issues by leveraging unlabeled text. These methods typically train a model on a small seed set of (instruction, answer) pairs to generate candidate instructions for unlabeled documents, which are then filtered to create additional training examples (Chen et al., 2024, 2023). While reducing reliance on extensive human labeling, these back-translation pipelines still critically hinge on an initial, manually-curated seed set.

This seed-dependency causes a core challenge: **the reliance on seed data inherently limits the full automation, diversity, and data efficiency**

*Corresponding author

of instruction tuning. Specifically, as illustrated in Figure 1, (i) assembling a diverse, high-quality seed set remains labour-intensive and costly; (ii) conditioning generation on a small seed corpus can transfer its stylistic and topical biases to the synthetic pairs, curbing diversity and generalisation; and (iii) the common practice of treating all unlabeled passages as answers can lead to data wastage, as question-formatted segments in raw corpora are often discarded or receive low-quality synthetic instructions. This raises a critical question: *How can we unlock the full potential of abundant raw text for instruction tuning, without the bottleneck of seed data, while ensuring high-quality and diverse instruction-following capabilities?*

To address this challenge, we propose CYCLE-INSTRUCT, a novel and fully **seed-free** framework for instruction tuning that requires no manually written seeds or external teacher models. Inspired by cycle consistency in unsupervised machine translation (He et al., 2016; Lample et al., 2018), CYCLE-INSTRUCT employs a dual self-training loop. We begin by automatically partitioning a large unlabeled corpus into *potential question passages* and *potential answer passages*, reformatted into instruction-tuning compliant `<instruction>` and `<response>` slots. An answer generator produces pseudo-responses for question passages, while a question generator back-translates pseudo-instructions for answer passages. These two models then act as mutual teachers: each learns to reconstruct the original passage by conditioning on the pseudo-label from its counterpart, using the reconstruction error as the training objective. Crucially, because every sample is supervised by its own **ground-truth content**, the learning signal closely approximates fully supervised training, allowing the models to internalise the true data distribution from the entire unlabeled corpus.

We thoroughly evaluate CYCLE-INSTRUCT across four diverse tracks. Our results consistently demonstrate that CYCLE-INSTRUCT generates coherent and relevant instruction-following data, establishing its potential as a scalable, especially when human-labeled resources are scarce or unavailable. Our main contributions are as follows:

- We propose CYCLE-INSTRUCT, a *fully seed-free* instruction-tuning framework that entirely eliminates reliance on human-written seeds and external teacher models, directly addressing the core bottleneck of current back-translation methods.
- We introduce a **dual self-training loop with cycle consistency** as the core mechanism, enabling two models to mutually supervise each other by reconstructing raw passages, yielding a high-quality learning signal akin to fully supervised training from unlabeled text alone.
- Through extensive experiments on four diverse data tracks, we demonstrate that CYCLE-INSTRUCT achieves performance **comparable to** strong supervised methods and significantly **outperforms** seed-driven back-translation baselines, all while requiring *zero* human annotations.

2 Background

2.1 Back-Translation for Instruction Tuning

Back-translation (BT) was introduced in NMT to exploit monolingual target data by "translating it back" into the source language (Sennrich et al., 2016). Recent work (Li et al., 2023a; Köksal et al., 2023) adapts this idea to instruction tuning.

Let $\mathcal{S} = \{(q, a)\}$ be a *small seed* set of question–answer pairs and $\mathcal{D}_A$ an unlabeled corpus of free-form answers.

1. **Seed training.** Train an *inverse model* $F_{A \to Q}$ to predict a question from an answer:

$$\mathcal{L}_{\text{inv}} = -\sum_{(q,a)\in\mathcal{S}} \log p_{\psi}(q \mid a). \tag{1}$$

2. **Pseudo-pair generation.** For each $a \in \mathcal{D}_A$, create a pseudo-question $\hat{q} = F_{A \to Q}(a)$ and form a synthetic pair $(\hat{q}, a)$.

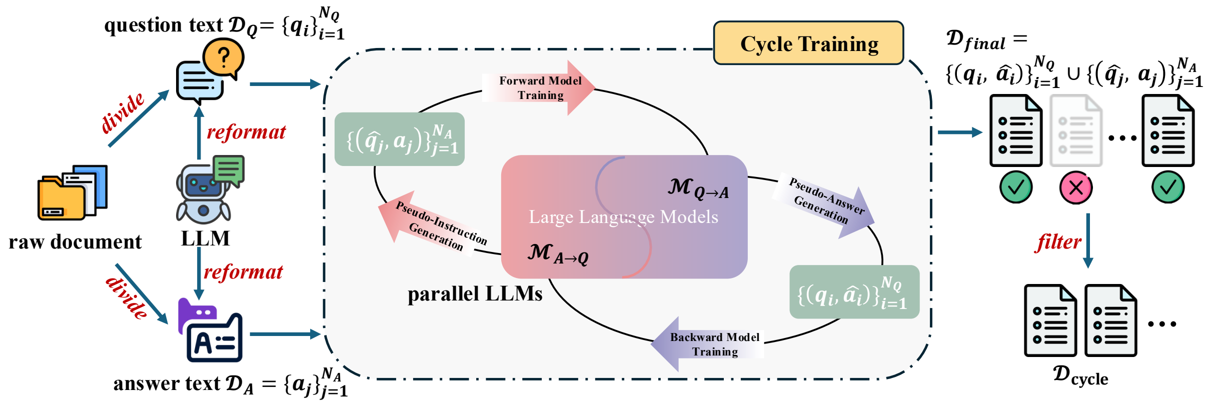
3. **BT fine-tuning.** Fine-tune the forward model $G_{Q \to A}$ on all synthetic pairs via

$$\mathcal{L}_{\text{BT}} = \mathbb{E}_{(\hat{q},a)}\big[-\log p_{\theta}(a \mid \hat{q})\big]. \tag{2}$$

The resulting model $G_{Q \to A}$ benefits from a much larger, automatically generated corpus, while all notation remains consistent with the cycle-consistency formulation that follows.

2.2 Cycle Consistency

Cycle consistency traces its roots to unsupervised machine translation and dual learning, where two models—one translating from language X to Y and the other from Y to X—are trained jointly by enforcing that translating "there and back" recovers



question text $\mathcal{D}_Q = \{q_i\}_{i=1}^{N_Q}$
divide
reformat
raw document
LLM
reformat
divide
answer text $\mathcal{D}_A = \{a_j\}_{j=1}^{N_A}$
Cycle Training
Forward Model Training
$\{(\hat{q}_j, a_j)\}_{j=1}^{N_A}$
Pseudo-Instruction Generation
$\mathcal{M}_{Q\to A}$
Large Language Models
$\mathcal{M}_{A\to Q}$
Pseudo-Answer Generation
$\{(q_i, \hat{a}_i)\}_{i=1}^{N_Q}$
parallel LLMs
Backward Model Training
$\mathcal{D}_{final} = \{(q_i, \hat{a}_i)\}_{i=1}^{N_Q} \cup \{(\hat{q}_j, a_j)\}_{j=1}^{N_A}$
filter
$\mathcal{D}_{\text{cycle}}$
a) Data Segmentation and Reformat
b) Cycle Training Procedure
c) Cycle-Consistency Filtering

Figure 2: The Overall Framework of our Fully Seed-Free Instruction Tuning Method.

the original sentence (He et al., 2016). Concretely, given mappings

$$G : X \to Y, \quad F : Y \to X, \tag{3}$$

one adds the reconstruction loss

$$\mathcal{L}_{\text{cycle}} = \mathbb{E}_{x\sim\mathcal{D}_X}\big[\ell\big(x,\, F(G(x))\big)\big] + \mathbb{E}_{y\sim\mathcal{D}_Y}\big[\ell\big(y,\, G(F(y))\big)\big], \tag{4}$$

where $\ell(\cdot,\cdot)$ is typically the negative log-likelihood of reconstructing the original sequence. CycleGAN (Chu et al., 2017) applied a similar cycle constraint in image translation by using ℓ_1 pixel losses, illustrating the concept across modalities.

In Cycle-Instruct, we reinterpret this mechanism for instruction tuning by viewing questions and answers as two analogous "languages." Let

$$G_{Q\to A} : \mathcal{Q} \to \mathcal{A}, \quad F_{A\to Q} : \mathcal{A} \to \mathcal{Q} \tag{5}$$

be the question-to-answer and answer-to-question models, respectively. We then minimize

$$\mathcal{L}_{\text{cycle}} = \mathbb{E}_{q\sim\mathcal{Q}}\Big[\ell\big(q,\, F_{A\to Q}\big(G_{Q\to A}(q)\big)\big)\Big] + \mathbb{E}_{a\sim\mathcal{A}}\Big[\ell\big(a,\, G_{Q\to A}\big(F_{A\to Q}(a)\big)\big)\Big]. \tag{6}$$

By enforcing $F_{A\to Q}(G_{Q\to A}(q)) \approx q$ and $G_{Q\to A}(F_{A\to Q}(a)) \approx a$ on unlabeled text, the two models effectively teach one another. This self-supervised reconstruction signal enables the generation of high-quality question–answer pairs directly from raw corpora, without any seed examples or external teachers.

3 Methodology

3.1 Overview

We propose CYCLE-INSTRUCT, a seed-free instruction tuning framework that leverages cycle-consistent dual-model training. Unlike previous instruction tuning methods requiring human-written seeds or external teacher models, CYCLE-INSTRUCT uses raw unlabeled text to iteratively bootstrap two LLMs: a forward model $\mathcal{M}_{Q\to A}$ and a backward model $\mathcal{M}_{A\to Q}$. The forward model generates pseudo-responses from extracted question passages, and the backward model back-translates pseudo-instructions from answer passages. Both models mutually reinforce each other's predictions, with training losses computed based on reconstruction of original data segments. Figure 2 illustrates the overall framework of our method.

3.2 Data Segmentation

We adopt an ultra-light, seed-free rule: a passage is a question iff it contains at least one question mark "?"; otherwise it is treated as an answer.

Concretely, each raw document $\mathcal{D}_{\text{raw}}$ is split into paragraphs by blank lines, and we obtain the raw paragraph sets

$$\mathcal{D}_Q^{\text{raw}} = \{\, q_i^{\text{raw}} \,\}_{i=1}^{N_Q}, \quad \mathcal{D}_A^{\text{raw}} = \{\, a_j^{\text{raw}} \,\}_{j=1}^{N_A}, \tag{7}$$

where q_i^{raw} (resp. a_j^{raw}) is a paragraph with (resp. without) a "?".

3.3 Data Reformat

To turn the split raw passages $\mathcal{D}_Q^{\text{raw}}$ and $\mathcal{D}_A^{\text{raw}}$ into INSTRUCTION–RESPONSE style data we apply two fixed rewriting prompts:

- **Prompter (questions).** For each $q_i^{\text{raw}} \in \mathcal{D}_Q^{\text{raw}}$ we ask the model to rewrite the paragraph into one self-contained, natural-sounding question q_i (See template in Appendix C.1).

- **Assistant (answers).** For each $a_j^{\text{raw}} \in \mathcal{D}_A^{\text{raw}}$ we ask the model to polish the text into a coherent answer paragraph a_j without introducing new information (See template in Appendix C.2).

The rewritten segments constitute the standardized datasets

$$\mathcal{D}_Q = \{\, q_i \,\}_{i=1}^{N_Q}, \quad \mathcal{D}_A = \{\, a_j \,\}_{j=1}^{N_A}, \qquad (8)$$

providing paired forms $(q_i,\ \_)$ and $(\_,\ a_j)$ that feed the four-step cycle training loop described latter.

3.4 Cycle Training Procedure

We instantiate two transformer models from the same base model (e.g., LLaMA3):

- Forward model: $\mathcal{M}_{Q\to A}(q; \theta_{Q\to A})$ generates responses given instructions.
- Backward model: $\mathcal{M}_{A\to Q}(a; \theta_{A\to Q})$ generates instructions given responses.

Here, $\theta_{Q\to A}$ and $\theta_{A\to Q}$ represent the parameters used by each model during generation, which are trained separately. However, under our self-training assumption, both models share the same base architecture.

Training proceeds iteratively in four cyclical steps:

Step 1: Pseudo-Answer Generation (using $\mathcal{M}_{Q\to A}$) : Given $\mathcal{D}_Q$, we generate pseudo-responses $\hat{a}_i$ (See template in Appendix C.3.):

$$\hat{a}_i = \mathcal{M}_{Q\to A}(q_i; \theta_{Q\to A}), \quad \forall q_i \in \mathcal{D}_Q \qquad (9)$$

resulting in pseudo-labeled pairs $\{(q_i, \hat{a}_i)\}_{i=1}^{N_Q}$.

Step 2: Backward Model Training (updating $\mathcal{M}_{A\to Q}$): Using pairs $(q_i, \hat{a}_i)$, we minimize the negative log-likelihood of reconstructing the original instructions:

$$\mathcal{L}_{A\to Q} = -\frac{1}{N_Q}\sum_{i=1}^{N_Q} \log P(q_i \mid \hat{a}_i; \theta_{A\to Q}) \qquad (10)$$

Step 3: Pseudo-Instruction Generation (using $\mathcal{M}_{A\to Q}$): Given $\mathcal{D}_A$, we generate pseudo-instructions $\hat{q}_j$ (See template in Appendix C.4.):

$$\hat{q}_j = \mathcal{M}_{A\to Q}(a_j; \theta_{A\to Q}), \quad \forall a_j \in \mathcal{D}_A \qquad (11)$$

producing pseudo-labeled pairs $\{(\hat{q}_j, a_j)\}_{j=1}^{N_A}$.

Step 4: Forward Model Training (updating $\mathcal{M}_{Q\to A}$): Using pairs $(\hat{q}_j, a_j)$, we minimize the negative log-likelihood of reconstructing the original responses:

$$\mathcal{L}_{Q\to A} = -\frac{1}{N_A}\sum_{j=1}^{N_A} \log P(a_j \mid \hat{q}_j; \theta_{Q\to A}) \qquad (12)$$

Iterative Refinement: We repeat Steps 1–4 iteratively, with each cycle progressively improving pseudo-label quality and better approximating the true underlying distribution of the unlabeled corpus.

Final Dataset Construction: After completing a fixed number of cycles T, we construct the final synthetic instruction-following dataset:

$$\mathcal{D}_{\text{final}} = \{(q_i, \hat{a}_i)\}_{i=1}^{N_Q} \cup \{(\hat{q}_j, a_j)\}_{j=1}^{N_A} \qquad (13)$$

This merged dataset combines pseudo-responses generated by $\mathcal{M}_{Q\to A}$ and pseudo-instructions generated by $\mathcal{M}_{A\to Q}$.

3.5 Cycle-Consistency Filtering (Optional)

Although $\mathcal{D}_{\text{final}}$ is already seed-free, we can further *audit and prune* its pseudo labels by checking cycle consistency under the final checkpoints $\theta_{Q\to A}^{(T)}$ and $\theta_{A\to Q}^{(T)}$.

(1) One-step reconstruction. For every pair in $\mathcal{D}_{\text{final}}$ we pass the *pseudo* side through the *opposite* model to obtain a reconstructed label:

$$\tilde{q}_i = M_{A\to Q}(\hat{a}_i; \theta_{A\to Q}^{(T)}), \ \ \tilde{a}_j = M_{Q\to A}(\hat{q}_j; \theta_{Q\to A}^{(T)}). \qquad (14)$$

(2) Embedding distance. We encode the original label and its reconstruction with the same sentence encoder $\phi(\cdot)$ (we use the LLaMA3 inference encoder without fine-tuning):

$$d_i = \|\phi(q_i) - \phi(\tilde{q}_i)\|_2, \ \ d_j = \|\phi(a_j) - \phi(\tilde{a}_j)\|_2. \qquad (15)$$

(3) Semantic clustering. To avoid domain or format bias in later pruning, we cluster the *gold sides*—$\{q_i\}$ and $\{a_j\}$—in the embedding space via k-means ($k = 200$ by default, tuned on a held-out slice). Each cluster thus represents a local region of the original data distribution.

(4) k-center greedy pruning. Within every cluster $\mathcal{C}$ we apply the *k-center greedy* selection rule to rank samples by distance score (d_i or d_j). We drop the top 5% farthest points:

$$\mathcal{C}_{\text{keep}} = \mathcal{C} \setminus \text{TopPercent}(\mathcal{C}, \text{dist}, 5\%). \qquad (16)$$

		Alpaca–GPT4					Dolly-15k				
Method	**Annot. (%)**	MMLU	BBH	CRASS	DROP	**Avg**	MMLU	BBH	CRASS	DROP	**Avg**
Vanilla	0	55.85	37.16	59.48	36.01	47.13	55.85	37.16	59.48	36.01	47.13
Random	5	57.92	38.38	68.98	38.06	50.84	55.42	37.14	62.41	33.79	47.19
	10	57.41	37.68	66.42	36.72	49.56	55.46	36.18	62.77	33.09	46.88
	20	57.63	37.23	67.88	37.39	50.03	55.91	36.51	63.50	33.06	47.25
Cluster	5	57.26	38.00	67.88	36.95	50.02	55.31	34.73	59.12	33.37	45.63
	10	57.57	37.68	68.61	36.75	50.15	55.48	35.83	61.68	33.57	46.64
	20	57.78	38.34	68.61	37.61	50.59	56.87	35.66	65.69	33.89	48.03
Cycle-Inst (Ours)	0	59.01	39.28	74.82	39.86	53.24	**58.96**	37.44	70.07	37.79	51.07
Cycle-Filt (Ours)	0	**59.39**	39.46	77.37	40.46	54.17	58.26	37.51	70.44	37.80	51.00
SFT-80	80	58.57	39.68	74.45	39.47	53.04	58.65	39.04	71.53	40.91	52.53
All-SFT	100	58.84	**40.53**	**79.20**	**41.37**	**54.99**	58.92	**39.72**	**74.09**	**41.05**	**53.45**

Table 1: Results on Alpaca–GPT4 and Dolly-15k for **Llama-3.1-8B**. Our methods surpass all back-translation baselines and approach the fully supervised **All-SFT** scores.

Because k-center greedy iteratively adds points that maximise the minimum pairwise distance, the retained set still covers the semantic support of $\mathcal{C}$ while discarding outliers that even the well-trained opposite model fails to reconstruct faithfully.

Resulting corpus. Concatenating pruned clusters yields

$$\mathcal{D}_{\text{cycle}} \subset \mathcal{D}_{\text{final}}, \qquad |\mathcal{D}_{\text{cycle}}| \approx 0.95\,|\mathcal{D}_{\text{final}}|. \tag{17}$$

We use $\mathcal{D}_{\text{cycle}}$ for all downstream supervised fine-tuning experiments.

4 Experimental Setup

4.1 Dataset Tracks and Baselines

We first test the proposed method on general instruction datasets. Next, we apply it to domain-specific datasets with limited labeled data. Finally and most importantly, following existing backtranslation works (Li et al., 2023a; Köksal et al., 2023; Chen et al., 2023), we conduct experiments on dialogue and plain text documents to showcase the effectiveness of our approach in generating coherent instruction-following data from raw text. For data in documents track, we leverage GPT-4o Mini to generate golden labels for the data, using the prompts provided in the Appendix C.3& C.4.

Track 1 — General instructions. Curated corpora such as **Alpaca–GPT4** (Peng et al., 2023) and **Dolly-15k**[1] provide human or GPT-4-authored instruction datasets, yet one side of the pair is frequently missing or unreliable.

[1]https://huggingface.co/datasets/databricks/databricks-dolly-15k

Track 2 — Domain-specific instructions. In specialist domains (e.g. medicine) aligned Q–A pairs are scarce. Using **Medical-Alpaca**[2]. we randomly remove one side of the pair, retaining 20 k unpaired questions and answers—emulating the common situation where only FAQ-style queries and unlabeled clinical notes exist.

Track 3 — Dialogue logs. Conversational logs preserve turn order but not explicit Q–A alignment. From **OASST-1**[3] we identify questions via a "?" heuristic (English + Chinese) and treat residual turns as answer candidates, yielding 15,126 Q and 24,874 A fragments.

Track 4 — Plain text. Narrative corpora embed latent questions within paragraphs. From 40 k **WikiHow**[4] articles we extract 5,178 interrogatives as potential instructions and label the remaining 34,822 sentences as answers, requiring both segmentation and synthetic alignment.

Baselines. We benchmark six settings—**Vanilla** (zero-shot), **All-SFT** (100 % gold pairs), **80%-SFT**, three seed-based back-translation variants(**Rand-k %**, which samples $k \in \{5, 10, 20\}$ % of the gold pairs uniformly at random, and **Clust-k %**, which chooses the same proportion from K-means clusters), and our seed-free **Cycle-Inst/Filt**. Full implementation details are provided in Appendix B.

[2]https://huggingface.co/datasets/medalpaca/medical_meadow_medical_flashcards

[3]https://huggingface.co/datasets/OpenAssistant/oasst1

[4]https://huggingface.co/datasets/wangwilliamyang/wikihow

Method	**Annot. (%)**	CK	CB	CC	CM	HB	HC	MG	PM	**Avg**
Vanilla	0	0.630	0.701	0.390	0.555	0.706	0.448	0.670	0.614	0.589
Random	5	0.657	0.715	0.410	**0.595**	0.719	0.453	0.720	**0.665**	0.617
	10	0.679	0.729	0.440	0.584	0.739	0.458	0.750	0.640	0.627
	20	0.679	0.708	0.440	0.584	0.748	0.478	**0.760**	0.662	0.632
Cluster	5	0.679	0.729	0.440	0.584	0.732	0.458	0.730	0.658	0.626
	10	0.657	0.736	0.430	0.566	0.745	0.468	0.750	0.629	0.623
	20	0.660	0.701	0.440	0.584	**0.752**	0.488	0.730	0.654	0.626
Cycle-Inst (Ours)	0	0.668	0.715	**0.460**	0.578	0.742	0.473	**0.760**	0.643	0.630
Cycle-Filt (Ours)	0	**0.687**	**0.757**	**0.460**	0.566	0.748	**0.483**	0.730	0.658	**0.636**
SFT-80	80	0.657	0.743	0.440	0.572	0.726	0.478	0.730	0.654	0.625
All-SFT	100	0.679	0.736	0.450	0.566	0.735	0.468	0.730	0.647	0.626

Table 2: MedAlpaca results on the nine medical sub-domains of MMLU for **Llama-3.1-8B**. Our seed-free methods outperform every back-translation baseline and even surpass the fully supervised **All-SFT**.

Dataset	**Pairs Used**	**Unlab. Q**	**Unlab. A**
Alpaca–GPT4	20,000	10,000	10,000
Dolly-15k	15,000	7,500	7,500
Medical-Alpaca	20,000	10,000	10,000
OASST-1 (logs)	40,000	15,126	24,874
WikiHow-4w (text)	40,000	5,178	34,822

Table 3: Statistics after subsampling. "Unlab." counts denote fragments lacking the opposite side of the pair and hence requiring synthesis.

4.2 Evaluation Protocol

Standard instruction metrics. For models trained on general instruction datasets, we follow `InstructEval` (Chia et al., 2023) and report accuracy (%) on MMLU (Hendrycks et al., 2020), BBH (Suzgun et al., 2022), CRASS (Frohberg and Binder, 2021), DROP (Dua et al., 2019). For the medical track we now follow the *eight* specialised sub-domains of the MMLU benchmark—CK (Clinical Knowledge), CB (College Biology), CC (College Chemistry), CM (College Medicine), HB (High-School Biology), HC (High-School Chemistry), MG (Medical Genetics), and PM (Professional Medicine).

Open-ended quality. We further report `AlpacaEval` (Li et al., 2023b) win-rate, the fraction of pairwise comparisons a system wins against the ALL-SFT baseline, providing a human-preference proxy for factual quality.

4.3 Implementation Details

All experiments fine-tune the **Llama-3.1-8B-Base** (Grattafiori et al., 2024) checkpoint with low-rank adaptation (LoRA) (Hu et al., 2022). For LoRA, we set the rank to 8, the scaling factor α to 16, and apply a dropout rate of 0.05. The learning-rate schedule is cosine decay from an initial value of 1×10^{-4}. Training employs a micro-batch size of 4, an effective batch size of 32, a sequence cutoff length of 1024 tokens, and runs for three epochs. All generation and evaluation are performed with vLLM (Kwon et al., 2023), using a maximum model length of 2048 tokens, top-k sampling with k = 10, a temperature of 0.2, and a generation limit of 500 tokens. Instruction-corpus experiments are trained on 8 × NVIDIA RTX 3090 (24 GB each), whereas raw-document experiments are trained on 8 × RTX 4090 (24 GB each).

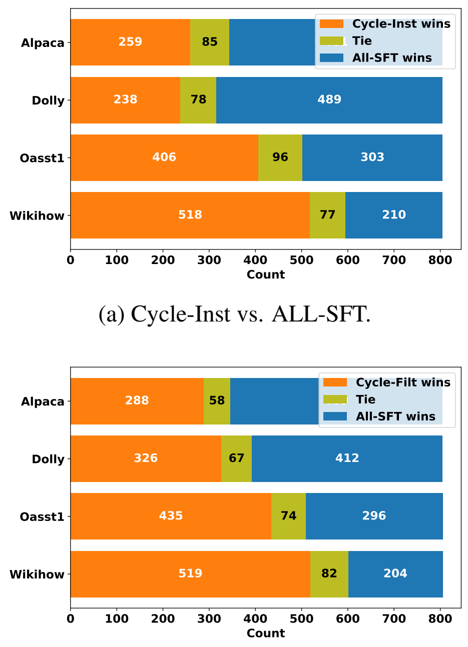
5 Results

5.1 Summary of Key Findings

1. Superior performance across all datasets. Our primary focus is on **raw-document-to-instruction-tuning** data synthesis, where our method establishes a new **SOTA**. On the nine medical MMLU sub-domains (Table 2), **OASST-1** and **WikiHow-4w** (Table 4), CYCLE-FILT achieves the highest scores among all methods. In each case it not only outperforms every seed-based back-translation baseline but also surpasses the fully supervised ALL-SFT model trained on 100% of the data, demonstrating clear advantages when labels are scarce or absent. On **Dolly-15k**, **Alpaca–GPT4** (Table 1), although our methods initially trail ALL-SFT (Table 1, Table 4), successive rounds of cycle-consistent pseudo-labeling allow CYCLE-FILT to match—or in some metrics even marginally exceed—the performance of the model trained on 100% of the data (for better models trained on **Dolly-15k** , see section 5.3). In every evaluation, our methods (CYCLE-INST and es-

		OASST-1 (Dialogue Logs)					WikiHow-4w (Plain Text)				
Method	**Annot. (%)**	MMLU	BBH	CRASS	DROP	**Avg**	MMLU	BBH	CRASS	DROP	**Avg**
Vanilla	0	55.85	37.16	59.48	36.01	47.13	55.85	37.16	59.48	36.01	47.13
Rand	5	56.99	37.48	65.69	36.99	49.29	57.64	36.59	68.61	34.72	49.39
	10	57.10	37.85	63.87	38.14	49.24	57.90	37.25	64.96	35.34	48.86
	20	57.38	36.32	70.44	38.87	50.75	57.86	37.29	68.25	35.13	49.63
Clust	5	57.24	37.76	68.61	38.27	50.47	57.02	36.29	63.05	35.67	48.01
	10	57.40	38.39	66.79	37.44	50.01	57.82	36.44	63.05	35.18	48.12
	20	57.69	38.40	67.15	38.09	50.33	57.49	36.72	65.69	34.91	48.70
Cycle-Inst (Ours)	0	58.77	38.75	**70.17**	39.19	51.72	58.49	37.54	67.88	38.57	50.62
Cycle-Filt (Ours)	0	59.07	**38.98**	70.07	**39.52**	**51.98**	58.70	**38.50**	67.52	**39.54**	**51.07**
SFT-80	80	58.81	38.26	68.61	37.80	50.87	58.28	37.55	70.07	36.47	50.59
All-SFT	100	**59.21**	38.56	70.07	38.92	51.69	**58.83**	37.76	**71.90**	35.40	50.97

Table 4: Results on OASST-1 and WikiHow for **Llama-3.1-8B**. Our methods beat all back-translation baselines and even achieve scores that exceed those of **All-SFT**.



(a) Cycle-Inst vs. ALL-SFT.

(b) Cycle-Filt vs. ALL-SFT.

Figure 3: Results on Alpaca Eval

pecially CYCLE-FILT) outperform the supervised model trained on 80% of the data. This holds true even when partial supervision is strong, underscoring the robustness of our synthetic-data approach.

2. Stability and benefits of clustered seed selection. Across seed budgets from 5 % to 20 %, CLUST-k variants deliver stable, monotonic gains on both general and raw-text tasks, whereas RAND-k variants show erratic performance—occasionally plateauing or degrading. This emphasizes both the importance and challenges of a diversity-based seed data selection mechanism while also highlighting the **limitations** of back-translation approaches that generate data based on seed-data training.

3. Effectiveness of cycle-consistency filtering. Filtering via cycle consistency (CYCLE-FILT) yields systematic improvements over the unfiltered CYCLE-INST across every benchmark, validating our hypothesis that noisy pseudo-pairs can be effectively removed through reconstruction verification.

4. Addressing back-translation shortcomings on multi-task instruction augmentation. All back-translation methods, initially underperform on the context-heavy Dolly-15k dataset (Table 1), trailing the fully supervised ALL-SFT model. We believe this stems from Dolly's multi-task instruction design: unlike purely declarative texts, its prompts specify explicit, diverse tasks that demand task-specific phrasing. Traditional back-translation focuses on matching answers to generated questions, yielding high QA alignment (see Table 5), but produces generic, one-size-fits-all instructions that lack the original task nuances (see Appendix D.1). In contrast, our iterative pseudo-labeling framework gradually infuses task specificity: each cycle generates instructions increasingly tailored to the correct task category, mitigating the generic drift inherent to back-translation (see Appendix D.2). By the second iteration, our synthetic instructions recover the complexity of Dolly's original prompts, significantly narrowing the performance gap and demonstrating robust adaptation even on complex, context-rich multi-task datasets (see Figure 4).

Alpaca Evaluation. Figure 3 presents the Alpaca-based evaluation results on four datasets,

	Random			Cluster			Cycle-Inst
Annot. (%)	5	10	20	5	10	20	0
Alpaca	9.21	8.99	9.09	9.17	9.14	9.31	**9.46**
Dolly	9.54	9.48	9.15	9.19	9.50	9.27	**9.90**
MedAlpaca	9.80	9.88	9.89	9.88	9.89	9.80	**9.96**
OASST-1	8.45	8.39	8.53	8.64	8.39	8.54	**8.75**
WikiHow	9.14	9.04	9.17	8.91	9.15	9.09	**9.43**

Table 5: GPT-4o mini evaluation of synthetic QA pair alignment.

Dataset	Pearson r	p-value
Alpaca	0.904	0.0052
Dolly	0.743	0.0559
MedAlpaca	0.646	0.117
OASST-1	0.872	0.0105
WikiHow	0.853	0.0146

Table 6: Pearson Correlation Coefficients and p-values between QA pair quality scores and downstream performance Across Datasets

comparing our two methods (CYCLE-INST, CYCLE-FILT) against the fully supervised ALL-SFT baseline. In the open-ended evaluation, we see the same pattern as before: our methods outperform the fully supervised ALL-SFT when converting raw documents into instructions. However, when it comes to augmenting a general instruction-tuning pool, we fall slightly behind ALL-SFT. Moreover, CYCLE-FILT consistently outperforms CYCLE-INST when pitted against ALL-SFT, further highlighting the superiority of our data-filtering strategy.

5.2 Synthetic Data Quality via GPT-4O MINI

GPT-4o Mini QA Pair Quality. To quantify how well each generated question matches its answer, we randomly sample 500 synthetic QA pairs per method and ask GPT-4O MINI to assign a single relevance score on a 0–10 scale (See Appendix C.5). Table 5 reports the average scores across four datasets. The results show that our methods yield substantially higher QA alignment than other back-translation baselines.

Correlation with Performance. Table 6 shows the Pearson correlation coefficients between GPT-4o mini QA pair quality scores and downstream performance for each method and dataset. We observe consistently high correlation values across all methods and corpora, indicating a strong positive relationship between QA alignment quality and task performance. This result further validates our core hypothesis: by leveraging unlabeled text to learn a broader, more diverse data distribution, we can synthesize tightly paired instruction–text examples that substantially improve the overall quality of the generated training data.

5.3 Understanding of the Cycle-Instruct's Iterative Refinement

Performance over Iterations. Figure 4 illustrates the change in performance for our methods over three rounds of iteration on each dataset (see

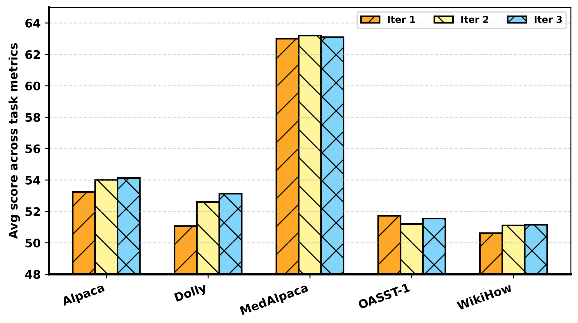


Figure 4: Cycle-Instruct's Performance over Iterations across different datasets.

Appendix E for more details). We observe that Dolly continues to improve steadily, Alpaca shows a modest uplift, whereas the other datasets plateau: MedAlpaca, OASST-1 and WikiHow remain essentially flat. As discussed in Section 5.1, this happens because on the simpler instruction datasets the model already generates sufficiently realistic pseudo-labels in the first training cycle, making further iterations unnecessary. Indeed, the strong performance gains seen after just one round in our main experiments further confirm that a single iteration is enough to capture the essential signal when task formats are straightforward.

6 Conclusion

We presented CYCLE-INSTRUCT, a fully seed-free instruction-tuning framework that relies on neither human-written seeds nor external teacher models. Through a dual self-training loop—an answer generator and a question generator linked by cycle consistency—the method extracts high-quality instruction–response pairs straight from raw text. Extensive experiments show that CYCLE-INSTRUCT consistently outperforms seed-driven back-translation baselines and delivers performance competitive with models trained on strong supervised data, underscoring its ability to tap vast unlabeled corpora for scalable, automated alignment of LLMs with human intent.

Limitations

First, due to resource constraints we only fine-tune model with LoRA, so the approach's behaviour at full-parameter or larger scales remains untested; second, the cycle-consistency objective could be exploited to reconstruct user prompts, posing potential privacy risks that call for defences such as differential privacy or rigorous red-teaming before deployment; finally, our seed-free segmentation relies on the presence of a question mark, which can fail on narrative or expository texts without explicit interrogatives, suggesting that richer discourse cues or retrieval-based heuristics should be explored.

Acknowledgments

This work is mainly supported by the National Key Research and Development Program of China (No. 2022YFB3304100). Haobo Wang is also supported by supported by the Fundamental Research Funds for the Zhejiang Provincial Universities (No. 226-2025-00004) and (No. 226-2024-00049).

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Shu Chen, Xinyan Guan, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Reinstruct: Building instruction data from unlabeled corpus. *arXiv preprint arXiv:2408.10663*.

Yongrui Chen, Haiyun Jiang, Xinting Huang, Shuming Shi, and Guilin Qi. 2023. Dog-instruct: Towards premium instruction-tuning data via text-grounded instruction wrapping. *arXiv preprint arXiv:2309.05447*.

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

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Casey Chu, Andrey Zhmoginov, and Mark Sandler. 2017. Cyclegan, a master of steganography. *arXiv preprint arXiv:1712.02950*.

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

Tianyu Dong, Bo Li, Jinsong Liu, Shaolin Zhu, and Deyi Xiong. 2025. Mlas-lora: Language-aware parameters detection and lora-based knowledge transfer for multilingual machine translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15645–15660.

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

Jörg Frohberg and Frank Binder. 2021. Crass: A novel data set and benchmark to test counterfactual reasoning of large language models. *arXiv preprint arXiv:2112.11941*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. *Advances in neural information processing systems*, 29.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2023. Longform: Optimizing instruction tuning for long text generation with corpus extraction. *arXiv preprint arXiv:2304.08460*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, and 1 others. Openassistant conversations-democratizing large language model alignment. corr, abs/2304.07327, 2023. doi: 10.48550. *arXiv preprint arXiv.2304.07327*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

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

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

Nihal Nayak, Yiyang Nan, Avi Trost, and Stephen Bach. 2023. Learning to generate instructions to adapt language models to new tasks. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, and 1 others. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Kai Tang, Junbo Zhao, Xiao Ding, Runze Wu, Lei Feng, Gang Chen, and Haobo Wang. 2024. Learning geometry-aware representations for new intent discovery. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5641–5654.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

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

Da Yin, Xiao Liu, Fan Yin, Ming Zhong, Hritik Bansal, Jiawei Han, and Kai-Wei Chang. 2023. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. In *EMNLP*.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. 2024. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE.

Shaolin Zhu, Leiyu Pan, Bo Li, and Deyi Xiong. 2024a. Landermt: Detecting and routing language-aware neurons for selectively finetuning llms to machine translation. *arXiv preprint arXiv:2409.19523*.

Shaolin Zhu, Shaoyang Xu, Haoran Sun, Leiyu Pan, Menglong Cui, Jiangcun Du, Renren Jin, António Branco, Deyi Xiong, and 1 others. 2024b. Multilingual large language models: A systematic survey. *arXiv preprint arXiv:2411.11072*.

Ingo Ziegler, Abdullatif Köksal, Desmond Elliott, and Hinrich Schütze. 2024. Craft your dataset: Task-specific synthetic dataset generation through corpus retrieval and augmentation. *arXiv preprint arXiv:2409.02098*.

A Related Work

A.1 Supervised Instruction Tuning

Supervised Instruction Tuning refers to fine-tuning large language models on explicit, human-written (or high-quality synthetic) instruction–response pairs that span a wide spectrum of tasks. Initial research focused on established NLP benchmarks and demonstrated that models fine-tuned on these task collections could generalize zero-shot to novel text-based problems (Wei et al., 2021; Sanh et al., 2021; Mishra et al., 2021; Tang et al., 2024). With the advent of ChatGPT and other conversational systems, the community shifted toward general-purpose instruction corpora that cover reasoning, coding, multimodal description, and dialogue, laying the groundwork for assistant-style LLMs (Ouyang et al., 2022; Chiang et al., 2023; Taori et al., 2023; Zhu et al., 2024b; Dong et al., 2025). In this supervised paradigm, the model first learns to map diverse prompts to high-quality answers, after which additional alignment steps such as RLHF or DPO can further refine helpfulness and safety (Ouyang et al., 2022; Rafailov et al., 2023). Crowd-sourced efforts such as OPENASSISTANT (Köpf et al.) and OPENHERMES (Conover et al., 2023) further democratise instruction data by collecting large human-annotated corpora. Despite strong performance, these approaches incur substantial annotation cost and remain constrained by the scope and biases of biases of available human prompts, motivating work on less supervised alternatives.

A.2 Backtranslation-Based Instruction Tuning

Instruction backtranslation generates synthetic instruction–response pairs for unlabeled text using a small seed set. Humpback (Li et al., 2023a) repeatedly augments web documents with seed-conditioned prompts and self-curates the best candidates. REINSTRUCT (Chen et al., 2024) adds lightweight passage filtering and answer rewriting. LONGFORM-C (Köksal et al., 2023) grounds answers in real documents before asking GPT-3 to author the corresponding instructions. DOG-INSTRUCT (Chen et al., 2023) wraps human-written documents into instruction form to curb hallucination while shrinking data size.

Although their pipelines and filtering strategies differ, all of these approaches ultimately derive data quality from the same backtranslation mechanism and thus inherit its common drawbacks: they still (i) depend on seed prompts or an external teacher model, (ii) suffer from data inefficiency because many synthetic pairs are low quality and must be filtered, and (iii) inherit distributional biases from the limited seed set, restricting instruction diversity. In contrast, CYCLE-INSTRUCT removes the seed bottleneck altogether: its dual self-training loop learns directly from raw corpora, needs no carefully selected seed data, and creates pseudo-pairs whose quality is enforced by cycle consistency rather than heavy post hoc filtering. Our experiments therefore focus on comparing the *generation paradigm* itself—seed-driven backtranslation versus fully seed-free cycle training. The results show that CYCLE-INSTRUCT consistently delivers higher downstream accuracy and human preference, confirming the advantages of seed-free, cycle-consistent data synthesis over traditional backtranslation pipelines.

A.3 Large-Model Validation

To further validate the effectiveness of our approach at scale, we fine-tune the **LLaMA 3.3 70B Instruct** model. The results are shown in Table 7. Interestingly, we observed that fine-tuning the model with the original dataset occasionally led to a drop in model performance. However, our approach—**Cycle-Filt** and **Cycle-Inst**—was able to maintain or even improve model performance, even when faced with a gap in data quality. This aligns with our previous observation that, as base models improve, **certain instruction data might not be sufficient to fine-tune these models effectively**. Our iterative self-distillation approach allows the model to continuously regenerate data and correct low-quality examples, thereby preserving and enhancing performance, which was not observed with traditional fine-tuning methods like All-SFT.

A.4 Comparisons with Seed-Data-Free Alternatives

Clarification on *seed-free*. In our setting, **seed-free** denotes generating instruction-following data *without manually curated seed examples and without external teacher models*. For context:

- **LongForm** (Köksal et al., 2023) introduces reverse instruction generation but relies on an *external teacher* to produce reverse instructions.

Method	OASST-1 (Dialogue Logs)					WikiHow-4w (Plain Text)				
	MMLU	BBH	DROP	CRASS	**Avg**	MMLU	BBH	DROP	CRASS	**Avg**
Vanilla	80.86	45.75	68.53	90.51	71.41	80.86	45.75	68.53	90.51	71.41
All-SFT	80.53	45.50	68.15	93.43	71.90	80.53	44.93	67.99	92.70	71.54
Cycle-Inst	**81.03**	**46.06**	**68.70**	**93.80**	**72.40**	**81.07**	**45.90**	**68.78**	**94.16**	**72.48**

Table 7: Large-model validation with **LLaMA 3.3 70B Instruct**. While ALL-SFT can regress relative to Vanilla, CYCLE-INST improves both OASST-1 and WIKIHOW-4W averages.

- **CRAFT** (Ziegler et al., 2024) uses *user-written* examples as demonstrations to model the target distribution; this depends on manually curated seeds.

- **Bonito** (Nayak et al., 2023) learns from a large pre-trained corpus; while partially seed-free, it may not faithfully match the target data distribution, yielding suboptimal averages in our replication.

We reproduce these methods under matched conditions (same base models and training parameters; *LongForm* updated to GPT-4o-mini for teacher-based steps) for a fair comparison. Table 8 reports the final results. CYCLE-FILT is competitive across both corpora and strongest on averages.

A.5 Additional Domain-Specific Results: CodeAlpaca & HumanEval

To broaden domain coverage, we additionally include **Code-Alpaca**[5] as a code-oriented instruction dataset and evaluate downstream code generation with **HumanEval** (Chen et al., 2021). This track assesses whether our seed-free synthetic instruction generation transfers to program synthesis tasks. Table 9 reports the final results. Results indicate that our method also carries over to code synthesis.

B Baseline Details

- **Vanilla.** Zero-shot performance of the base model (0 % labels).

- **All-SFT.** Supervised fine-tuning on 100 % gold instruction–response pairs.

- **80%-SFT.** Supervised fine-tuning on 80 % of the gold pairs.

- **Rand-k % / Clust-k %.** To model the scarcity of seed examples, we sample only $k \in \{5, 10, 20\}$ % of the gold pairs—uniformly at random (**Rand**) or via K-means clustering in embedding space (**Clust**). The observed side of each seed pair is *back-translated* to create its missing counterpart, and the model is fine-tuned on the resulting synthetic corpus. Varying k explores the trade-off between seed size and performance.

- **Cycle-Inst / Cycle-Filt.** Our one-round, seed-free framework with 0 % labels. **Cycle-Inst** trains directly on all synthetic pairs produced by the dual self-training loop, whereas **Cycle-Filt** further prunes pairs whose reconstructions violate a 5 % k-center-greedy cycle-consistency threshold (see Section 3.5 for more details).

[5] `https://huggingface.co/datasets/sahil2801/CodeAlpaca-20k`

C Prompt Templates

C.1 Prompt template for `REFORMAR_PROMPTER`

Figure 5 presents the template used by the `REFORMAR_PROMPTER`. Given a web passage that already contains question-like sentences, it prompts the model to craft one well-formed, natural question that could plausibly be asked about (part of) that passage, without collapsing the entire text into a summary. This question generation step seeds the subsequent answer-rewriting stage.

C.2 Prompt template for `REFORMAR_ASSISTANT`

Figure 6 presents the template used by the `REFORMAR_ASSISTANT`. Here, the "assistant" is instructed to rewrite that passage into a direct answer that is fluent, coherent, and preserves the original structure—laying the groundwork for a clean *(question, answer)* pair.

C.3 Prompt template for `Pseudo-Answer Generation`

Figure 7 depicts the template used when we already have an *instruction* (or question) but need a synthetic answer. The model is cast as an assistant asked to provide a helpful, high-quality response,

Method	OASST-1					WikiHow-4w				
	MMLU	BBH	DROP	CRASS	**Avg**	MMLU	BBH	DROP	CRASS	**Avg**
Bonito	58.02	38.47	39.38	67.52	50.85	56.49	37.96	39.76	61.31	48.88
CRAFT	58.84	38.90	38.62	70.07	51.61	58.31	38.60	38.12	67.15	50.55
LongForm	57.75	37.30	38.54	58.76	48.09	57.81	36.29	36.44	62.04	48.15
Cycle-Inst	58.77	38.75	39.19	70.17	51.72	58.49	37.54	38.57	67.88	50.62
Cycle-Filt	**59.07**	**38.98**	**39.52**	**70.07**	**51.91**	**58.70**	**38.50**	**39.54**	**67.52**	**51.07**

Table 8: Reproduced comparisons with seed-data-free alternatives. CYCLE-FILT is competitive across both corpora and strongest on averages.

REFORMAR_PROMPTER

Below is a block of Web text containing several paragraphs with question marks. Based on the content provided (or a portion of it), generate one plausible and clear question without summarizing the entire text. Maintain a natural, interrogative tone.
Web text: {instruction}
Answer:

Figure 5: Prompt template for `REFORMAR_PROMPTER`

Method	HumanEval
Cluster 5%	24.39
Cluster 10%	25.00
Cluster 20%	26.83
Random 5%	21.95
Random 10%	21.34
Random 20%	23.17
Normal 80%	26.22
Normal 100%	28.05
Cycle-Inst	**28.05**
Cycle-Filt	**28.66**

Table 9: HumanEval results with **CodeAlpaca** as the domain-specific dataset. CYCLE-FILT reaches the best performance among compared settings.

allowing us to bootstrap an answer in the absence of human annotations.

C.4 Prompt template for `Pseudo-Instruction Generation`

Figure 8 inverts the previous step: starting from a model-generated answer, it prompts the system to reconstruct a plausible *instruction* that would elicit that answer. This back-translation closes the loop, enabling us to create balanced instruction–answer pairs even when only one side was originally available.

C.5 Prompt template for `GPT4OMINI_QA_Evaluator`

Finally, Figure 9 illustrates the automated evaluation prompt. Given a synthetic *(Q, A)* pair, GPT-4o-mini is asked to return a single relevance score from 0 – 10, enabling large-scale filtering of low-quality pairs without manual review.

D Cases

D.1 Dolly Failure Cases

Figure 10 highlights how traditional back-translation succeeds on declarative web texts but fails on Dolly's context-rich, multi-task prompts, producing generic instructions that lose task specificity.

D.2 Iterative Instruction Refinement Cases for Dolly-15k

Figure 11 illustrates a classification example from the Dolly-15k dataset, contrasting the generic instruction produced by traditional back-translation with the progressively refined instructions generated by our Cycle-Filt framework over two iterations. This visualization demonstrates how iterative pseudo-labeling restores the original task-specific phrasing.

E Iteration-wise Performance for Cycle-Inst

Below we report Table 10 and Table 11, for each of five datasets, the Cycle-Inst method's metrics over three self-training iterations.

REFORMAR_ASSISTANT

A single-turn chat between a curious user and an artificial intelligence assistant. The assistant gives helpful answers to the user's questions.
User: Below is a block of Web text without any question marks. Please rewrite it into a fluent and coherent response that clearly conveys its intended meaning. The overall structure should remain similar, but ensure the language flows smoothly and the purpose is unmistakable.
Web text: {output}
Assistant:

Figure 6: Prompt template for `REFORMAR_ASSISTANT`.

Pseudo-Answer Generation

A single-turn chat between a curious user and an artificial intelligence assistant. The assistant gives helpful answers to the user's questions.
User: {instruction}
Assistant:

Figure 7: Prompt template for `Pseudo-Answer Generation`.

Pseudo-Instruction Generation

Below is a reponse from an AI Assistant and its user instruction. The instruction is used as prompt for the response.
Assistant: {output}
User:

Figure 8: Prompt template for `Pseudo-Instruction Generation`.

GPT4OMINI_QA_Evaluator

You are an AI evaluator. For a given Answer (A) and Generated Question (Q), score the relevance of the question to the answer on a scale from 0 to 10, where 0 means completely irrelevant and 10 means perfectly relevant. Respond ONLY with a single numeric score.
A: {answer}
Q: {question}
Relevance score (0–10):

Figure 9: Prompt template for `GPT4OMINI_QA_Evaluator`.

Other datasets:
"golden_answer": "Cans or tins can be repurposed to protect young seedlings from snails and other pests. This is a simple and effective solution."
"golden_instruction": "How can I protect my seedlings from snails and other pests?"
"generated_instruction": "How can I protect my seedlings from snails and other pests?"

Dolly datasets:
"task_category": "classification"
"golden_answer: "Human: Two Legs\nHorse: Four Legs\nDog: Four Legs\nCat: Four Legs\nMonkey: Two Legs\nKangaroo: Two Legs\nBoar: Four Legs"
"golden_instruction": "Classify each of the following as having two or four legs: human, horse, dog, cat, monkey, kangaroo, boar"
"generated_instruction": "Write a program that prints the number of legs of a given animal."

Figure 10: Unlike declarative texts in other datasets, Dolly's multitask-based prompts embed explicit tasks, so the heightened variability and compatibility issues between prompts make accurate inversion particularly difficult and often lead to the generation of low-quality instructions.

"task_category": "classification"
"golden_answer": "Human: Two Legs\nHorse: Four Legs\nDog: Four Legs\nCat: Four Legs\nMonkey: Two Legs\nKangaroo: Two Legs\nBoar: Four Legs"
"golden_instruction": "Classify each of the following as having two or four legs: human, horse, dog, cat, monkey, kangaroo, boar"
"backtranslation_generated_instruction": "Write a program that prints the number of legs of a given animal."
"Cycle in iter1_generated_instruction": "Extract the nouns and verbs from the following sentence. Separate them with a comma. The dog ran, the cat swam, John ate, Adam flew."
"Cycle in iter2_generated_instruction": **"Classify the following words as either nouns or verbs: dog, run, cat, eat, swim, John, Adam, Fly."** ☑

Figure 11: Iterative pseudo-instruction refinement for a classification example from Dolly-15k, showing the back-translation output, Cycle-Instruct iteration 1, and Cycle-Instruct iteration 2, progressively recovering the multi-instance classification task wording.

Dataset	**Iteration**	**MMLU**	**BBH**	**CRASS**	**DROP**	**Avg**
Alpaca–GPT4	Iter 1	59.01	39.28	74.82	39.86	53.24
	Iter 2	**59.39**	39.23	**77.37**	**40.05**	54.01
	Iter 3	59.01	**40.14**	**77.37**	40.02	**54.13**
Dolly-15k	Iter 1	58.96	37.44	70.07	37.79	51.07
	Iter 2	59.11	38.06	74.82	**38.41**	52.60
	Iter 3	**59.13**	**38.41**	**76.64**	38.36	**53.13**
OASST-1	Iter 1	58.77	**38.75**	**70.17**	39.19	**51.72**
	Iter 2	**59.15**	37.87	69.50	38.28	51.20
	Iter 3	58.65	38.09	**70.17**	**39.28**	51.55
WikiHow	Iter 1	**58.49**	37.54	67.88	**38.57**	50.62
	Iter 2	58.29	**38.37**	70.07	37.72	51.11
	Iter 3	58.33	37.19	**70.80**	38.27	**51.15**

Table 10: Iteration-wise performance on four general datasets using Cycle-Inst.

Iteration	**CK**	**CB**	**CC**	**CM**	**HB**	**HC**	**MG**	**PM**	**Avg**
Iter 1	**0.668**	0.715	**0.460**	**0.578**	0.742	0.473	**0.760**	0.643	0.630
Iter 2	0.657	0.722	0.440	**0.578**	0.755	0.478	0.750	0.673	**0.632**
Iter 3	0.638	**0.729**	0.440	0.566	**0.758**	**0.483**	**0.760**	**0.676**	0.631

Table 11: Iteration-wise performance on Medical-Alpaca using Cycle-Inst.