Aligning Instruction Tuning with Pre-training

Yiming Liang [*1 2 3] **Tianyu Zheng** [*4 5] **Xinrun Du** [*4 5] **Ge Zhang** [*4] **Xingwei Qu** [4] **Xiang Yue** [4] **Chujie Zheng** [5]
Jiaheng Liu [4] **Lei Ma** [3 6] **Wenhu Chen** [4] **Guoyin Wang** [5] **Zhaoxiang Zhang** [2] **Wenhao Huang** [4] **Jiajun Zhang** [1 2]

Abstract

Instruction tuning enhances large language models (LLMs) to follow human instructions across diverse tasks, relying on high-quality datasets to guide behavior. However, these datasets, whether manually curated or synthetically generated, are often narrowly focused and misaligned with the broad distributions captured during pre-training, limiting LLM generalization and effective use of pre-trained knowledge. We propose *Aligning Instruction Tuning with Pre-training* (AITP), a method that bridges this gap by identifying coverage shortfalls in instruction-tuning datasets and rewriting underrepresented pre-training data into high-quality instruction-response pairs. This approach enriches dataset diversity while preserving task-specific objectives. Evaluations on three fully open LLMs across eight benchmarks demonstrate consistent performance improvements with AITP. Ablations highlight the benefits of adaptive data selection, controlled rewriting, and balanced integration, emphasizing the importance of aligning instruction tuning with pre-training distributions to unlock the full potential of LLMs.

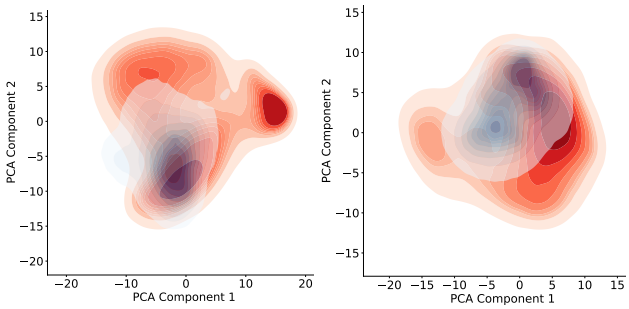

Figure 1: **Visualization of Projections**. The red regions at the bottom represent the pre-training corpus, while the light blue regions above represent the SFT datasets. Darker areas indicate a higher concentration of data points, whereas lighter areas represent sparser distributions. Additional projections are shown in Appendix A.

1. Introduction

Instruction tuning is essential for adapting large language models (LLMs) to effectively follow human instructions across diverse tasks. This process relies on high-quality datasets to guide model behavior, yet existing instruction-tuning datasets are often narrowly focused, relying on either manual annotation or synthetic generation. While manual datasets offer precision, they are costly and lack scalability (Wang et al., 2022b; Zhou et al., 2023a). Synthetic datasets, on the other hand, frequently depend on expensive APIs of strong models and are tightly coupled with their generation pipelines, limiting flexibility (Peng et al., 2023; Lian et al., 2023). Additionally, manually combining open-source datasets, as seen in efforts like OpenHermes-2.5 (Teknium, 2023) and Tulu-V2 (Ivison et al., 2023), often overlooks the underlying data distributions, leading to inefficiencies.

Pre-training corpora, by contrast, reflect broader real-world distributions and align closely with the internal knowledge of LLMs, making them a rich source of high-quality supervisory signals. However, current instruction-tuning methods fail to leverage this alignment, creating a fundamental gap in optimizing dataset coverage and distribution. Addressing this challenge requires aligning instruction-tuning datasets with pre-training distributions to fully exploit the knowledge embedded in LLMs.

In this paper, we propose Aligning Instruction Tuning with Pre-training (AITP), a method that systematically bridges this gap. Rather than generating instruction-response pairs from scratch, AITP identifies gaps in existing datasets by comparing their distribution to that of the pre-training corpus. Underrepresented data is then rewritten into high-quality instruction-response pairs, enhancing dataset coverage and alignment. As shown in Figure 2, AITP involves three stages: (1) generating a difference set based on density comparisons, (2) rewriting raw text into instruction-response

[*]Equal contribution [1]School of Artificial Intelligence, University of Chinese Academy of Sciences [2]Institute of Automation, Chinese Academy of Sciences [3]BAAI [4]M-A-P [5]01.ai [6]Peking University. Correspondence to: Ge Zhang <gezhang@umich.edu>, Jiaheng Liu <buaaljiaheng@gmail.com>, wenhaohuang <wenhaohuang@xxx.edu>, JiaJun Zhang <jjzhang@nlpr.ia.ac.cn>.

Preprint. Work in Progress

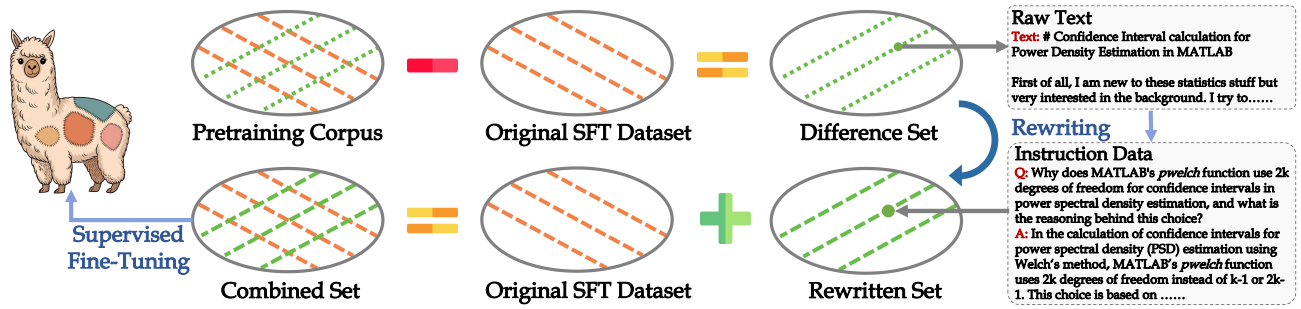

Figure 2: **The pipeline of AITP.** AITP first generates a difference set, then rewrites the raw text into instruction-response pairs to form a rewritten set, and finally combines the rewritten set with the original SFT dataset for model training.

pairs, and (3) integrating these pairs into the original dataset for fine-tuning.

Figure 1 visualizes the significant distributional differences between instruction-tuning datasets and the pre-training corpus, underscoring the need for such alignment. Through experiments on three open-source LLMs across eight benchmarks, we demonstrate that AITP consistently improves model performance. Detailed ablation studies highlight the effectiveness of adaptive data selection and integration, showing how AITP guides instruction tuning toward more effective and generalizable fine-tuned models.

Our contributions include: 1) Demonstrating the distributional gaps between instruction-tuning datasets and pre-training corpora through visualization. 2) Proposing the AITP method to adaptively optimize instruction-tuning datasets by leveraging pre-training corpora as a reference. 3) Validating the effectiveness of AITP with extensive experiments and ablation studies.

2. Methods

2.1. Difference Set Generation

In this section, we define the process of difference set generation, isolating data points from the pre-training corpora that differ from those in the SFT dataset. The goal is to identify regions in the pre-training data distribution that are absent from or sparsely populated in the supervised fine-tuning (SFT) data. This can be formalized as follows:

$$D_{\text{diff}} = \{d_i | d_i \in D_{\text{pretrain}}, \Delta(d_i, D_{\text{SFT}}) < \tau\} \quad (1)$$

Where D_{pretrain}, D_{SFT}, D_{diff} represent the pre-training dataset, the SFT dataset and the resulting difference set, respectively. $\Delta(d_i, D_{\text{SFT}})$ represents the density estimate of the data point d_i in the SFT dataset, and τ is the threshold that determines whether a data point should be included in the difference set. To achieve this, we outline the procedure in three main stages: data representation, density estimation, and identification of the difference set.

2.1.1. Data Representation

Each data point is represented as a vector derived from the final-layer embedding of the model. We then apply dimensionality reduction (DR) to project these high-dimensional embeddings into two-dimensional coordinates, facilitating visualization and density comparison across datasets. This process can be formalized as follows:

$$(x_i, y_i) = \text{DR}(\text{Model}(d_i)) \quad (2)$$

Applying the same dimension reduction to both pre-training and SFT embeddings results in two sets of two-dimensional vectors:

$$Z_{\text{pretrain}} = \{(x_i, y_i) \mid d_i \in D_{\text{pretrain}}\} \quad (3)$$
$$Z_{\text{SFT}} = \{(x_i, y_i) \mid d_i \in D_{\text{SFT}}\} \quad (4)$$

2.1.2. Density Estimation

To compare data distributions between the pre-training and SFT datasets, we use Kernel Density Estimation (KDE) to visualize the density of points for each dataset. The KDE function $\hat{f}(x, y)$ estimates the density at any location (x, y) based on neighboring points:

$$\hat{f}(x, y) = \frac{1}{n h_x h_y} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_x}, \frac{y - y_i}{h_y}\right) \quad (5)$$

$K(\cdot, \cdot)$ is the kernel function, typically Gaussian:

$$K((x, y), (x\prime, y\prime)) = \exp\left(-\frac{(x-x\prime)^2 + (y-y\prime)^2}{2\sigma^2}\right) \quad (6)$$

Where (x, y) and $(x\prime, y\prime)$ are two two-dimensional data points, h_x, h_y and σ are bandwidth parameters that control the smoothness in the x direction, y direction and kernel respectively. The KDE visualization highlights distribution differences, identifying regions of divergence between the pretraining and SFT datasets.

2.1.3. FINDING DIFFERENCE SET

The difference set is identified based on the density estimates from the SFT dataset. Specifically, if a point d_i in the pre-training dataset has a low-density estimate within the SFT dataset, we classify this point as absent or sparsely populated in the SFT data. Such points contribute to the observed distributional differences between the two datasets, and we define them formally as:

$$D_{\text{diff}} = \{d_i | d_i \in D_{\text{pretrain}}, \hat{f}_{\text{SFT}}(x_i, y_i) < \tau\} \quad (7)$$

$\hat{f}_{\text{SFT}}(x_i, y_i)$ represents the density estimate of the data point d_i from the pretrain corpus within the SFT dataset.

$$\hat{f}_{\text{SFT}}(x_i, y_i) = \frac{1}{nh_x h_y} \sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h_x}, \frac{y_i - y_j}{h_y}\right) \quad (8)$$

Where $(x_i, y_i) \in Z_{\text{pretrain}}$, $(x_j, y_j) \in Z_{\text{SFT}}$. n is the total number of points in the SFT dataset.

2.2. Data Transformation of Difference Set

The data transformation phase is designed to convert raw text from pre-training data within the difference set into instruction-pair data formatted for SFT. First, we develop a query generation prompt to guide the model in generating relevant questions from the raw text. Next, we implement a query scoring prompt to assess the quality of each generated query. Low-quality queries are filtered out based on these scores, enabling us to eliminate unsuitable questions before answer generation, thus conserving computational resources. Finally, an answer generation prompt is applied to instruct the model in generating responses to the remaining high-quality queries.

2.3. Training

In this phase, the model is trained on a combined dataset that includes both the rewritten data derived from the difference set and the original SFT dataset. Notably, the model trained on the combined dataset is the same as the one trained on the pre-training corpus. This serves two main purposes: first, it ensures consistency between the supplemented knowledge distribution and the model's internal knowledge. Second, high-quality instruction-pair data helps correct semantic inaccuracies that may arise from formatting errors in the pre-training corpus.

3. Experiment Settings

3.1. Evaluation

We evaluate the model's instruction-following ability using the IFEval benchmark (Zhou et al., 2023b), which is unbiased because it does not rely on LLM-generated evaluation scores. It provides four types of accuracy scores: Prompt-level Strict-accuracy (P-S), Instruction-level Strict-accuracy (I-S), Prompt-level Loose-accuracy (P-L), and Instruction-level Loose-accuracy (I-L). We use the OpenCompass, a comprehensive, one-stop platform for LLM evaluation (Contributors, 2023). We evaluate the effectiveness of AITP across seven standard benchmarks. These benchmarks provide a comprehensive evaluation of the diverse capabilities of language models across various tasks and domains. MMLU (Hendrycks et al., 2021) offers a broad assessment of multitask reasoning and knowledge retrieval, while ARC-c (Clark et al., 2018) and GPQA-diamond (Rein et al., 2023) focus on complex scientific reasoning and physics-specific understanding, respectively. For code generation and problem-solving, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) measure a models ability to write correct and multi-step logical solutions. Additionally, HellaSwag (Zellers et al., 2019) tests commonsense reasoning by predicting contextually appropriate continuations, and GSM8K (Cobbe et al., 2021) challenges models with elementary-level math problems, combining natural language understanding with mathematical reasoning.

3.2. Main Setting

Our experiments utilize three fully open-source models: OLMo (Groeneveld et al., 2024), MAP-Neo (Zhang et al., 2024a) and Pythia (Biderman et al., 2023). These models not only release model weights but also training datasets and intermediate checkpoints, aiming to facilitate reproduction and advance scientific research in LLMs. In this paper, the OLMo-7B-base, MAP-Neo-7B-base, and Pythia-12B models, along with their corresponding pre-training corpora, are chosen as the foundational setup for AITP. The OLMo-7B-SFT and MAP-Neo-7B-SFT-v0.1 models are used as baselines to validate the effectiveness of AITP. Since the SFT dataset for Pythia has not been released, we use Tulu-v2 for fine-tuning as the baseline for Pythia.

Due to the substantial storage and computational resources required for the data embedding and shift phase, we don't use the full pre-training corpus given resource constraints. Instead, we apply reservoir sampling (Vitter, 1985), an algorithm that enables uniform sampling from streaming data, ensuring that the sampled subset maintains a distribution consistent with the full pre-training corpus. The reservoir sampling algorithm is described in the Appendix B.

We conduct experiments on the NVIDIA A800-SXM4-80GB, with the difference set generation phase taking approximately 56 GPU hours. The Data Transformation Setting phase utilizes the vLLM (Kwon et al., 2023) framework to accelerate inference, requiring approximately 640 GPU hours, while the Training Setting phase, involving full-parameter fine-tuning, takes approximately 256 GPU