LEAD: Iterative Data Selection for Efficient LLM Instruction Tuning

Xiaotian Lin
HKUST (GZ)
xlin420@connect.hkust-gz.edu.cn

Yanlin Qi
Université Paris Cité
yanlinqi7@gmail.com

Yizhang Zhu
HKUST (GZ)
yzhu305@connect.hkust-gz.edu.cn

Themis Palpanas
Université Paris Cité
themis@mi.parisdescartes.fr

Chengliang Chai
BIT
ccl@bit.edu.cn

Nan Tang
HKUST (GZ)
nantang@hkust-gz.edu.cn

Yuyu Luo*
HKUST (GZ)
yuyuluo@hkust-gz.edu.cn

ABSTRACT

Instruction tuning has emerged as a critical paradigm for improving the capabilities and alignment of large language models (LLMs). However, existing iterative model-aware data selection methods incur significant computational overhead, as they rely on repeatedly performing full-dataset model inference to estimate sample utility for subsequent training iterations. In this paper, we propose LEAD, a framework that LEArns to select Data iteratively by accurately estimating sample utility entirely within the standard training loop, eliminating the need for additional model inference. At its core, LEAD introduces Instance-Level Dynamic Uncertainty (IDU), a theoretically grounded utility function combining instantaneous training loss, gradient-based approximation of loss changes, and exponential smoothing of historical loss signals. To further scale efficiently to large datasets, LEAD employs a two-stage, coarse-to-fine selection strategy, adaptively prioritizing informative clusters through a multi-armed bandit mechanism, followed by precise fine-grained selection of high-utility samples using IDU. Extensive experiments across four diverse benchmarks show that LEAD significantly outperforms state-of-the-art methods, improving average model performance by 6.1%-10.8% while using only 2.5% of the training data and reducing overall training time by 5-10×.

PVLDB Reference Format:
Xiaotian Lin, Yanlin Qi, Yizhang Zhu, Themis Palpanas, Chengliang Chai, Nan Tang, and Yuyu Luo. LEAD: Iterative Data Selection for Efficient LLM Instruction Tuning. PVLDB, 19(3): 426 - 439, 2025.
doi:10.14778/3778092.3778103

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/HKUSTDial/LEAD.

1 INTRODUCTION

Instruction tuning improves large language models (LLMs) by fine-tuning on instruction–response pairs [2, 12, 37, 60, 75], aligning them with user intents and task formats to improve their generalization to diverse tasks such as data preparation [69], question answering [32–35], and data analysis [44, 48–52]. Beyond scale, data quality is the primary driver of gains [2, 74], motivating automatic selection of informative subsets using diversity or quality metrics [4, 11, 55, 66, 72]. However, such metric-only approaches ignore model feedback and cannot adapt to the model's evolving state. In response, *model-aware* selection leverages model-derived signals, either in a single shot (non-iterative) or across multiple rounds (iterative) [59, 63]. Non-iterative methods select data once based on initial model predictions, but their effectiveness is limited as they do not adapt to model evolution during training [37, 67, 68].

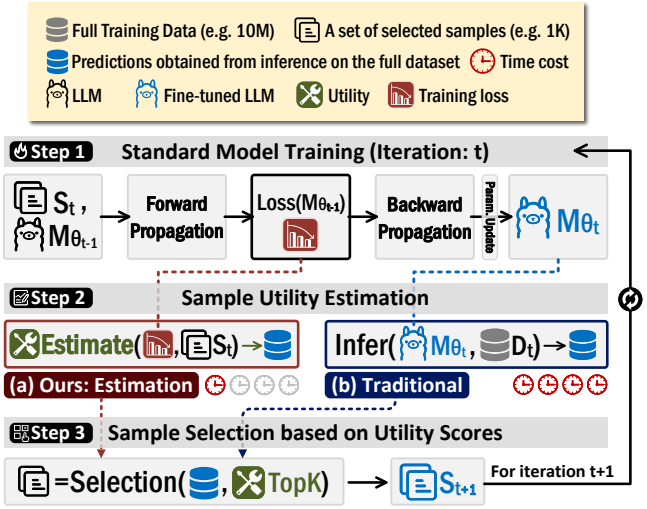

Figure 1: Comparison of Iterative Model-Aware Solutions. Here, S_t denotes the selected training subset at iteration t, and D_t denotes the full training dataset.

In contrast, iterative model-aware data selection methods follow a three-step loop (Figure 1). Within each iteration t, the process unfolds as follows: Starting with selected samples (S_t) and model (M_{t-1}) previously obtained in last iteration $t-1$, **Step 1** fine-tunes the model M_{t-1} on S_t to produce an updated model M_t; **Step 2** estimates sample utility across the dataset using newly obtained model M_t feedback (typically through full-dataset inference); and **Step 3** selects the next subset S_{t+1} based on these utilities. This process repeats, with the newly created S_{t+1} and M_{t+1} serving as inputs to iteration $t+1$.

*Yuyu Luo is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 19, No. 3 ISSN 2150-8097.
doi:10.14778/3778092.3778103

As shown in Figure 1-Step 2-(b), most existing iterative model-aware methods rely on explicit model inference to assess the utility of samples. Specifically, after each training iteration, these methods perform inference on *every* sample in the training set to derive feedback signals (*e.g.,* model uncertainty scores) for utility estimation. For example, the IFD [37] requires approximately 98 GPU-hours to select data from a pool of $600K$ samples in a single round.

This predicament leads to a natural **research question**: *Can we retain the benefits of iterative model-aware data selection without repeatedly performing costly full-dataset inference?*

In this work, we posit that the answer is yes. As shown in Figure 1-Step 1, our key insight is that during standard training, the model first conducts a forward propagation step using the current mini-batch of samples, computes the per-sample losses based on its predictions, and subsequently updates its parameters via backward propagation. Crucially, this training process naturally produces a per-sample loss for each training instance in the mini-batch. Intuitively, this loss indicates how challenging a sample is for the model. *If we can cleverly harness these inherent training signals across the whole dataset*, we could **estimate** the utility of each sample **without additional inference (inference-free)** (see Figure 1-Step 2-(a)). This idea – leveraging training-time loss signals to guide data selection – offers the potential to eliminate the full-dataset inference stage while still adapting to the model's training state. However, realizing this idea in practice is ***challenging***.

First, although using training-time losses allows us to avoid explicit inference, a subtle yet fundamental issue arises due to a timing misalignment. Specifically, as shown in Figure 1-Step 1, the training loss observed at iteration t reflects the model's performance *before* updating parameters (model state $M_{\theta_{t-1}}$), whereas the utility of selecting samples ideally should consider their usefulness *after* the parameter update (*i.e.,* M_{θ_t} at iteration $t+1$). This temporal mismatch means that naively reusing pre-update loss signals may not accurately reflect true sample utility after the next parameter update. We term this issue as the *temporal mismatch* challenge (**C1**).

Second, raw loss signals can be noisy or unstable – they fluctuate from one update to the next due to randomness (*e.g.,* varying batch composition) and the non-stationary nature of training, thus naively trusting instantaneous loss values might lead to suboptimal choices. This issue highlights the *instability of loss signals* challenge (**C2**).

Third, even if we eliminate separate inference steps, individually estimating utility and selecting informative samples remains inefficient for large-scale datasets (*e.g.,* containing millions of samples). We denote this as the *sample-level selection efficiency challenge* (**C3**).

Our Methodology. We propose LEAD, a *theoretically grounded* data selection framework that integrates the data selection process into the model training loop and estimates sample utilities without additional inference overhead (*i.e., inference-free*). To achieve this, we first propose a sample utility estimation function called *Instance-Level Dynamic Uncertainty* (**IDU**). IDU explicitly implements the `Estimate` step depicted in Figure 1-Step 2-(a) by combining three naturally available training signals: (1) the current training loss for each sample, (2) gradient-based approximation, derived from gradient correlation approximations, to anticipate loss changes at the next parameter update (addressing **C1**), and (3) historical loss trends via exponential smoothing to reduce random noise and improve stability (addressing **C2**). Importantly, IDU is computed entirely using training-time signals naturally available during model updates (losses and logits), thus incurring no additional inference overhead. Finally, we derive the Lagrangian function and utilize complementary slackness conditions to determine the optimal parameters for IDU rigorously. Guided by this theoretical foundation, our LEAD framework employs a practical coarse-to-fine data selection strategy (Figure 2).

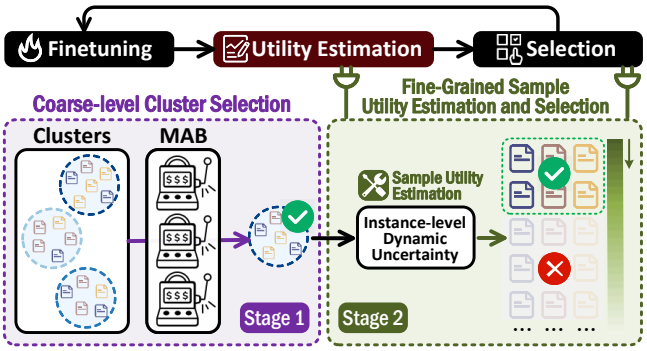

Figure 2: A High-level Overview of LEAD.

Stage 1: Coarse-level Cluster Selection. Recall our third challenge (**C3**) – efficient candidate selection at scale. To address this, we first partition the dataset offline into clusters based on two widely-used metrics: (1) *instruction-following difficulty*, measuring how challenging each instruction is for the model [37], and (2) *task-level similarity*, grouping semantically related instructions [39]. This clustering step is performed only once per dataset. During training, LEAD employs a multi-armed bandit (MAB) algorithm [62] to dynamically identify and prioritize clusters likely to yield higher rewards – clusters containing samples with greater potential to significantly enhance the model's performance (addressing **C3**).

Stage 2: Fine-Grained Sample Utility Estimation and Selection. Within each selected cluster, LEAD utilizes the IDU function to estimate the utility of individual samples precisely. Specifically, given the IDU scores computed based on the previously discussed training signals (losses, historical trends, and gradient predictions), LEAD prioritizes and selects samples with the highest IDU values.

Contributions. This paper makes the following contributions:

(1) Problem Formulation. We formally introduce the problem of Iterative Data Selection with Inference-Free Utility Estimation, defining a scenario where iterative model-aware selection is performed without incurring additional inference overhead (Section 2).

(2) Instance-Level Dynamic Uncertainty (IDU). We propose a new sample utility estimation function, IDU. It addresses the temporal mismatch and instability of loss signals by combining current losses, gradient-based approximation of loss changes, and exponential smoothing of historical loss signals, all computed using naturally available training signals without extra model inference (Section 3).

(3) LEAD Framework. We propose LEAD, a theoretically grounded and efficient iterative data selection framework seamlessly integrated into the standard model training process, eliminating repeated costly inference steps (Section 4 and Section 5).

(4) Theoretical Analysis. We rigorously ground our framework in a Lagrangian optimization formulation, employing complementary slackness conditions and gradient correlation approximations to derive theoretically optimal parameters for the IDU function, ensuring both soundness and practical effectiveness (Section 6).

(5) Extensive Experiments. Extensive experiments across four diverse benchmarks show that LEAD significantly outperforms state-of-the-art methods, improving average model performance by 6.1%-10.8% while using only 2.5% of the training data and reducing overall training time by 5-10× (Section 7).

2 PRELIMINARY & PROBLEM FORMULATION

2.1 Data Selection for Instruction Tuning

Instruction tuning adapts a pretrained LLM $\mathcal{M}_\theta$ to follow instructions by fine-tuning on instruction–response pairs $(x, y) \in \mathcal{D}$: $\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[L(\mathcal{M}_\theta(x), y) \right]$, where L is a task-specific loss function such as cross-entropy.

Static Data Selection for Instruction Tuning. Given a dataset $\mathcal{D}$, it selects a fixed subset $\mathcal{D}^* \subseteq \mathcal{D}$ under budget constraint B: $\min_{\mathcal{D}^*\subseteq\mathcal{D},\ |\mathcal{D}^*|\le B} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{target}}} \left[L(\mathcal{M}_\theta(x), y) \right]$, where $\mathcal{D}_{\text{target}}$ denotes the target distribution. However, static methods cannot adaptively select samples based on the model's evolving capabilities to maximize learning effectiveness during training [2].

Iterative Data Selection for Instruction Tuning. Iterative data selection interleaves model fine-tuning and data selection across multiple iterations. Formally, given the model parameters θ_t at iteration t, we adaptively select a subset $S_t \subseteq \mathcal{D}$ based on a utility function $f(\theta_t, x)$, which estimates the expected contribution of each sample x to future model improvement (*e.g.,* loss reduction). The iterative selection problem can thus be formulated as:

$$\max_{\{S_1,\ldots,S_T\}} \sum_{t=1}^{T} \sum_{x\in S_t} f_t(\theta_t, x), \quad \text{s.t.} \quad \sum_{t=1}^{T} |S_t| \le B, \tag{1}$$

where B is the total sample selection budget allowed during training.

Existing methods typically estimate the utility $f_t(\theta_t, x)$ by performing full-dataset inference at each iteration. Specifically, after fine-tuning the model on selected samples S_t, traditional methods explicitly run inference on the entire dataset $\mathcal{D}$ using the updated model parameters θ_t to compute utility scores:

$$f_t(\theta_t, x) = g(\text{Infer}(\theta_t, x)), \quad \forall x \in \mathcal{D}, \tag{2}$$

where $\text{Infer}(\theta_t, x)$ denotes inference (*e.g.,* loss or uncertainty computation) and $g(\cdot)$ maps inference results to utility values.

Consequently, the next subset S_{t+1} is selected as:

$$S_{t+1} = \underset{S_t\subseteq\mathcal{D},\ |S_t|\le k}{\arg\max} \sum_{x\in\mathcal{D}} f_t(\theta_t, x), \quad \text{s.t.} \quad |S_t| \le k, \quad T\cdot k \le B. \tag{3}$$

Note that in iterative data selection, we typically assume a fixed selection size k per iteration, constrained by the total selection budget B. Thus, the number of iterations T and the selection size per iteration k satisfy the relation $T \cdot k \le B$.

2.2 Problem Formulation

*Definition 2.1 (**Iterative Data Selection with Inference-Free Utility Estimation**). Given a total sample selection budget B, our objective is to identify subsets $\{S_t\}_{t=1}^T$ that maximize the cumulative estimated utility, where the utility function $f_t(\theta_{t-1}, x)$ is computed exclusively from training-time signals (e.g., training losses or logits) without incurring additional inference overhead:*

$$\max_{\{S_1,\ldots,S_T\}} \sum_{t=1}^{T} \sum_{x\in S_t} f_t(\theta_{t-1}, x), \quad \text{s.t.} \quad \sum_{t=1}^{T} |S_t| \le B, \tag{4}$$

Specifically, at each iteration t, the utility estimation $f_t(\theta_{t-1}, x)$ utilizes the loss signal computed using model parameters θ_{t-1} immediately after the forward propagation step, but before the backward propagation (parameter update). Thus, no additional inference is required to estimate utilities for data selection at iteration t.

Our goal, therefore, is to design accurate and stable inference-free utility estimation methods. For simplicity, we use $f_t(\theta_{t-1}, x)$ and $f(\theta_{t-1}, x)$ interchangeably when the context clearly refers to data selection at iteration t.

3 INSTANCE-LEVEL DYNAMIC UNCERTAINTY UTILITY

Designing an effective inference-free utility function $f(\theta_{t-1}, x)$ requires addressing two fundamental challenges as discussed in Section 1: **(C1)** the temporal mismatch between pre-update loss signals and their actual post-update utility, and (**C2**) the instability of instantaneous loss signals due to random fluctuations and noise.

To tackle these challenges, we first define a baseline utility function based on a *loss-based uncertainty metric*, and then introduce an improved formulation, termed *Instance-Level Dynamic Uncertainty (IDU)* utility function, which explicitly addresses these limitations.

Loss-based Uncertainty Estimation. Specifically, our approach begins by formalizing Instance-level uncertainty through a loss-based formulation. Formally, given an instruction-response pair (x, y), we define the Instance-level Uncertainty (IU) [22] at training iteration t as the empirical cross-entropy between the model's current predictive distribution and the ground-truth response:

$$IU(\theta_t, y|x) = L(\theta_t, x) = -\frac{1}{T}\sum_{j=1}^{T} \log p_{\theta_t}(t_j^y | x, t_1^y, \ldots, t_{j-1}^y), \tag{5}$$

where T is response length, t_j^y refers to the j-th response token, and p_{θ_t} the model's token-level predictive probability distribution.

IU naturally corresponds to the training-time negative log-likelihood loss, providing a direct and computationally free baseline. However, IU alone cannot effectively handle challenges **(C1)** and **(C2)**.

Instance-Level Dynamic Uncertainty. To explicitly mitigate both temporal mismatch (**C1**) and instability (**C2**) of loss signals, we introduce the Instance-Level Dynamic Uncertainty (IDU), which incorporates exponential smoothing of historical losses and gradient-based approximation of loss changes. Formally, given subset S_t at iteration t, IDU for sample x is recursively defined as:

$$\begin{aligned} f(\theta_{t-1}, x) &= IDU(\theta_{t-1}, x) \\ &= (1-b)\cdot \underbrace{\Big[\underbrace{L(\theta_{t-1}, x)}_{\text{IU at } \theta_{t-1}} + \underbrace{\Delta L'(\theta_t, x)}_{\text{Utility Change}}\Big]}_{\textbf{Estimated} \text{ Utility at } \theta_t} + b \cdot \underbrace{IDU(\theta_{t-2}, x)}_{\text{Historical Utility}}, \end{aligned} \tag{6}$$

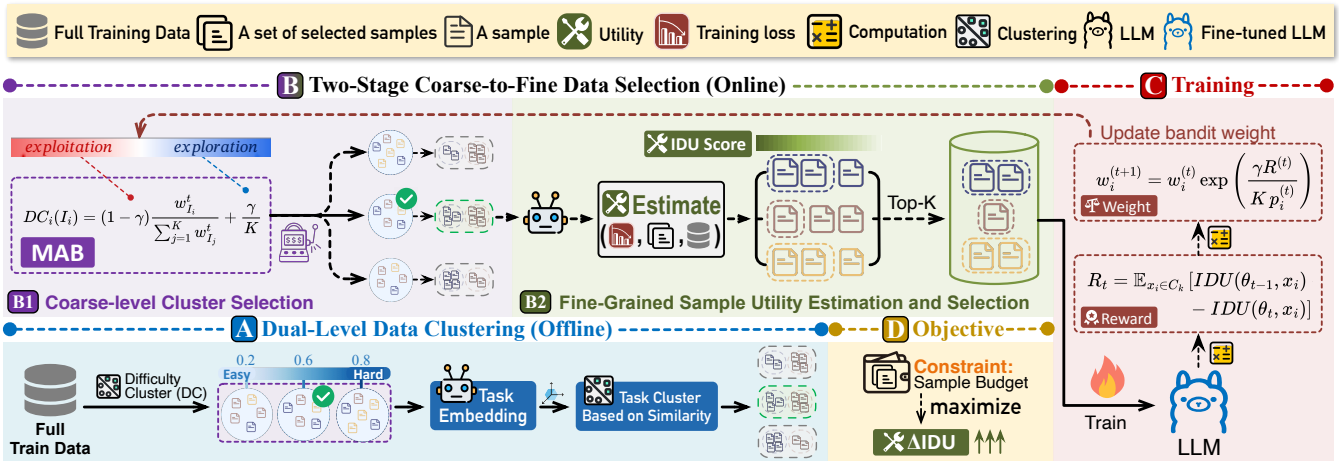

Figure 3: An Overview of the LEAD Framework.

where $b \in [0, 1)$ controls the balance between current and historical signals, $L(\theta_{t-1}, x)$ is the IU computed using model parameters θ_{t-1}, and $\Delta L'(\theta_t, x)$ is an approximation of the expected utility change, defined as: $\Delta L'(\theta_t, x) = L(\theta_t, x) - L(\theta_{t-1}, x)$.

We have the following key clarifications regarding Eq. (6):

- The instantaneous loss $L(\theta_{t-1}, x)$ is computed naturally during forward propagation at iteration t, requiring no extra inference.
- The $\Delta L'(\theta_t, x)$ denotes the anticipated loss change from θ_{t-1} to θ_t. Importantly, this estimation leverages only readily available gradient and historical loss information collected at iteration $t-1$, ensuring no extra inference is performed at iteration t.

IDU effectively resolves both fundamental challenges through two carefully designed components:

- **Utility Change Estimation (Gradient-Based approximation).** To address temporal mismatch (**C1**), IDU explicitly estimates the expected utility change ($\Delta L'(\theta_t, x)$) between consecutive iterations. Instead of performing additional inference passes with updated parameters (θ_t), we leverage gradient-based approximations derived from backward propagation at iteration $t-1$ to estimate the loss at iteration t.
- **Historical Utility (Exponential Smoothing).** To tackle instability (**C2**), IDU incorporates historical uncertainty signals using an exponential smoothing mechanism. Rather than depending solely on instantaneous IU values, IDU maintains an exponential moving average of previous utility estimates ($IDU(\theta_{t-2}, x)$). This significantly reduces fluctuations caused by random noise and local minima encountered during training.

We will elaborate on the details of computing IDU and optimizing the coefficient b of the IDU utility function in Section 5.1.

4 LEAD: LEARNING-TO-SELECT DATA ITERATIVELY

We first present an overview of LEAD (Section 4.1), followed by the three key components enabling inference-free iterative data selection (Section 4.2). Finally, we describe how these components systematically interact during iterative training (Section 4.3).

4.1 LEAD Framework: An Overview

LEAD adopts a *coarse-to-fine* strategy. It first performs a one-time *dual-level data clustering* to cluster data by difficulty and task similarity. During training, an online selector combines a Multi-Armed Bandit (MAB) scheduler with our IDU function, enabling model-aware selection without repeated full-dataset inference.

Dual-Level Data Clustering (Offline). As shown in Figure 3-(A), we first partition the dataset into clusters based on two dimensions: instruction-following difficulty [37] and task similarity [39].

The goal is to align the training data with the model's evolving capability while maintaining task diversity. Concretely, at each training iteration, we first select a difficulty-level cluster that best matches the model's current learning capacity. This ensures that the model is always trained on samples of appropriate challenge, facilitating stable and efficient learning progression. Within the selected difficulty cluster, we further sample across different task clusters proportionally. This allows the model to be exposed to a diverse set of tasks at the same difficulty level, helping improve generalization and prevent overfitting to narrow task types. This dual-level clustering is conducted offline, incurring no additional computational overhead during online training.

(1) Difficulty-aware Instance-level Clustering. To align training samples with the model's current capability, we use the Instruction-Following Difficulty (IFD) [37], a widely adopted metric [36, 43, 68] for quantifying conditional complexity of instructions, to evaluate instance-level difficulty. IFD evaluates instance-level difficulty by comparing how challenging an instruction is for the model. Additionally, PPL [36], a fundamental metric in language modeling, quantifies how well a model predicts a sample, with lower perplexity indicating a better and more confident prediction. The IFD score leverages perplexity to assess the difficulty of instructions relative to the model's capacity. Formally, given an instruction-response pair (x, y), the IFD is computed as: $IFD(y \mid x) = \frac{PPL(y|x)}{PPL(y)}$, where $PPL(y \mid x)$ and $PPL(y)$ denote the perplexities of generating the y with and without the x, respectively. Using these IFD scores, we group training samples into clusters through kmeans algorithm

and use the silhouette coefficient to determine the optimal number of clusters k.

(2) Similarity-based Task-level Clustering. Within each difficulty cluster, we further conduct finer-grained clustering based on task similarity to encourage the model to learn diverse task types of comparable difficulty, similar to how students study multiple subjects within the same grade. Specifically, we extract task-specific embeddings from instructions by emphasizing task-defining terms (*e.g.,* key verbs and nouns), following the approach in [39]. We then apply the K-means [52] to group instructions by task similarity.

Coarse-to-Fine Data Selection (Online). During the training, as shown in Figure 3-(B), LEAD implements a coarse-to-fine selection process designed to maximize utility and training effectiveness under a given total sample budget.

(1) Coarse-Level Cluster Selection (via MAB). At each training iteration t, we first employ a Multi-Armed Bandit (MAB) algorithm (specifically EXP3, detailed in Section 5.2) as a coarse-level scheduler to guide the exploration process. Its role is to dynamically prioritize one difficulty-level cluster that is most beneficial to the current model state. The MAB algorithm leverages a self-guided IDU-based reward signal, directly measuring the reduction in IDU scores derived from training on previously selected clusters, allowing the model to adaptively focus on the most beneficial cluster for the subsequent fine-grained selection.

(2) Fine-Grained Sample Selection (via IDU). After identifying the optimal difficulty-level cluster, we distribute the selection budget across its finer-grained task clusters. Specifically, we select the most informative samples from each task cluster based on their current IDU values (see Section 5.1), thus ensuring efficient fine-grained selection of training data at iteration t.

These selected samples form the subset S_t used to fine-tune the model at iteration t. After training, the model parameters are updated from θ_{t-1} to θ_t, and the MAB rewards are updated accordingly, ensuring the LEAD framework continuously improves its data selection strategy.

4.2 LEAD Framework: Core Components

(1) Instance-Level Dynamic Uncertainty (IDU) Utility. To estimate sample utility efficiently without additional inference, we introduce the Instance-Level Dynamic Uncertainty (IDU) metric. IDU combines exponential smoothing of historical losses and a gradient-based approximation of loss change, effectively addressing the temporal instability and inference overhead challenges inherent in traditional iterative selection methods (see Section 5.1).

(2) Adaptive Data Selection via MAB-Integrated Training Scheduler. To integrate coarse and fine-grained selections seamlessly, we employ the EXP3 algorithm to dynamically balance exploration and exploitation among clusters. The MAB scheduler dynamically prioritizes clusters demonstrating higher historical utility gains, thus efficiently adapting to the model's evolving learning capabilities (further described in Section 5.2).

(3) Self-Guided IDU-Based Reward. To guide the coarse-level cluster selection via MAB, we propose a novel reward function based on the reduction of IDU achieved by training on a given cluster without the need for external validation steps and additional inference (Please refer to Section 5.3 for details).

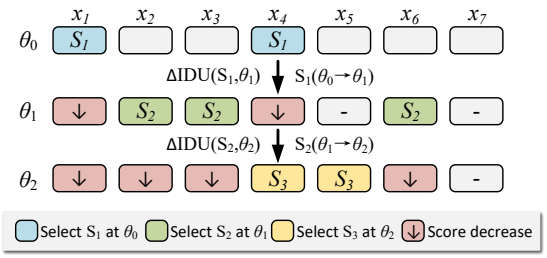

Figure 4: Iterative Sample Selection Guided by IDU Scores.

4.3 Training Iteration Workflow of LEAD

The LEAD integrates iterative data selection with LLM instruction tuning. Each training iteration t within LEAD comprises four steps.

Step 1: Difficulty-Aware Cluster Selection. Select the optimal coarse-level difficulty cluster C_{i^*} via the MAB-EXP3 algorithm, guided by the reward derived from previous training iterations, reflecting the cluster's historical effectiveness.

Step 2: Fine-Grained Sample Selection. Within the cluster C_{i^*}, utilize the IDU function to select the top n_{i^*} most informative samples. These samples form the training subset S_t. For example, in Figure 4, at iteration θ_0, samples with the highest initial IDU scores (labeled as S_1) are chosen for training.

Step 3: LLM Instruction Tuning. The selected samples (S_t) are used to fine-tune the model parameters, transitioning from the current parameters θ_{t-1} to the updated parameters θ_t.

Step 4: Reward and Utility Updates. After fine-tuning, trained samples typically show decreased IDU scores, reflecting reduced informativeness. This reduction serves as the training reward. As shown in Figure 4, lowered IDU scores of previously selected samples (*e.g.,* S_1 at θ_0 and S_2 at θ_1) prompt dynamic selection of new, more informative samples for subsequent iterations (*e.g.,* S_2 to S_3). Finally, both IDU scores and the MAB weights are updated accordingly, guiding the sample selection process in future iterations.

Through this structured workflow, LEAD continuously and adaptively selects the most beneficial samples at each training step.

5 THE DESIGN DETAILS OF LEAD

We first show how to optimize our IDU utility under a budget constraint (Section 5.1), followed by an adaptive data selection scheduler via MAB algorithms (Section 5.2), and finally, a self-guided IDU-based reward for cluster evaluation (Section 5.3).

5.1 Instance-Level Dynamic Uncertainty Optimization under the Budget Constraint

In Section 3, we introduced the *IDU* utility (Eq. (6)) for estimating sample utilities in iterative data selection. Note that our LEAD aims to iteratively select subsets of samples with the highest cumulative utility gain, defined as the expected reduction in average *IDU* at each iteration (ΔIDU_t) under a total budget constraint B. Formally, our optimization problem can be defined as follows.

Problem 1 (Budget-Constrained IDU Utility Optimization). *Given a total selection budget B, our goal is to maximize the cumulative expected utility over T training iterations:*

$$\max_{b,T} \sum_{t=1}^{T} \mathbb{E}[\Delta IDU_t], \quad s.t. \sum_{t=1}^{T} \mathbb{E}[n_t] \leq B \tag{7}$$

$$where \quad \mathbb{E}[n_t] = \alpha \cdot (1-b) \cdot |\overline{C}| \cdot (1 + CV^2) \cdot (1 + O(\gamma)) \tag{8}$$

Here, n_t denotes the number of samples selected at iteration t, α is the sampling ratio, $b \in [0, 1)$ is the smoothing parameter controlling the influence of historical utility, $|\overline{C}|$ is the average cluster size, and $CV^2 = \frac{1}{K}\sum_{i=1}^{K} \frac{(|C_i| - |\overline{C}|)^2}{|\overline{C}|^2}$ quantifies variability among cluster sizes.

To solve this problem, we construct a Lagrangian function incorporating the budget constraint and apply the complementary slackness condition to derive the optimal smoothing parameter b^*. Specifically, the optimal smoothing coefficient b^* that maximizes cumulative utility gain under the budget constraint is given by: $b^* = 1 - \frac{B}{\alpha \cdot |\overline{C}| \cdot T \cdot (1+\mathrm{CV}^2)}$. The detailed derivation and theoretical justification of b^* are provided in Theorem 6.1 (Section 6).

In practice, to effectively implement the optimal solution to our budget-constrained utility maximization problem, we first derive the optimal smoothing coefficient b^* from the theoretical analysis above. However, to fully instantiate our IDU utility function, we must also efficiently estimate the utility changes ($\Delta L'(\theta_t, S_t)$) between consecutive training iterations, as this term directly contributes to computing the cumulative utility gain ΔIDU_t. Directly calculating these utility changes would typically require additional inference steps, violating our zero-cost constraint.

To address this, we introduce the gradient-based approximation of utility change, as discussed below.

Gradient-Based Approximation of Utility Change. Our approach efficiently utilizes gradient information computed during standard model training, thus requiring no extra computational resources beyond regular forward-backward propagation.

Formally, consider a subset of samples S_i. When model parameters are updated from θ_{t-1} to θ_t, the average uncertainty change (utility change) $\Delta L(\theta_t, S_i)$ can be approximated as follows:

Theorem 5.1 (Utility Change Approximation). *For a given sample subset S_i, the utility change from parameter update θ_{t-1} to θ_t can be approximated as:*

$$\begin{aligned}\Delta L'(\theta_t, S_i) &\equiv \frac{1}{|S_i|} \sum_{x \in S_i} (L(\theta_t, x) - L(\theta_{t-1}, x)) \\ &\approx -\eta \left[\beta^2 \delta_{t_k} + (1-\beta)^2 \delta_{t-1} + 2\beta(1-\beta)\sqrt{\delta_{t_k}\delta_{t-1}} \cos\phi\right]\end{aligned} \tag{9}$$

where η is the learning rate, δ_{t_k} and δ_{t-1} denote historical gradient norms, and ϕ is the angle between consecutive gradient directions, given by: $\cos\phi = \frac{\Delta\theta_{t_k}^\top \Delta\theta_{t-1}}{\|\Delta\theta_{t_k}\| \cdot \|\Delta\theta_{t-1}\|}$.

This approach ensures that our utility estimation remains efficient, accurate, and fully integrated into standard model training workflows. The complete derivation of this gradient-based approximation method is presented in Theorem 6.4 (Section 6).

While the above approximation method significantly enhances efficiency, its accuracy critically depends on selecting an appropriate approximation coefficient β. To further refine our method, we analytically derive the optimal approximation weight β^* that minimizes approximation error.

Optimal Approximation Coefficient β^*. Formally, we define the approximation error function as: $J(\beta) = \|\Delta L(\theta_t, S_i) - \Delta L'(\theta_t, S_i)\|^2$. Minimizing this error function leads us to the theoretical β^*:

Theorem 5.2 (Optimal Weight β^*). *The optimal approximation weight β^* minimizing the error function $J(\beta)$ is given by:*

$$\beta^* = \frac{\delta_{t-1} - \sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi}{\delta_{t_k} + \delta_{t-1} - 2\sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi}. \tag{10}$$

Detailed proofs and analyses regarding the derivation of this optimal coefficient are provided in Theorem 6.4 (Section 6).

Finally, to rigorously evaluate the theoretical guarantees and practical utility of our gradient-based approximation, we establish a formal approximation error bound as follows.

Approximation Error Bound. We bound the approximation error between the approximated loss L' and the true loss L.

Theorem 5.3 (Approximation Error Bound). *With the optimal weight β^*, the error between the approximated loss L' and the true loss L satisfies:*

$$\|L'(\theta_t, x) - L(\theta_t, x)\| \leq \epsilon_{taylor} + \epsilon_{approx}\,,$$

where:

- $L'(\theta_i, x) = L(\theta_{i-1}, x) + \Delta L'(\theta_t, S_t)$
- $\epsilon_{taylor} = \frac{1}{2}\eta^2 \cdot \max_\theta \|\nabla^2 L(\theta, x)\| \cdot \|\nabla L(S_i, \theta_{i-1})\|^2$ *is the error from Taylor expansion.*
- $\epsilon_{approx} = \eta \cdot \|\nabla L(S_i, \theta_{i-1}) - (\beta^* \cdot \nabla L(S_{i_k}, \theta_{i_k-1}) + (1-\beta^*) \cdot \nabla L(S_{i-1}, \theta_{i-2}))\|^2$ *is the error from gradient approximation.*

5.2 Adaptive Data Selection via MAB-Integrated Training Scheduler

In this section, we propose a novel training scheduler for the LEAD framework that integrates the Multi-Armed Bandit (MAB) algorithm with our IDU utility function. The scheduler adaptively selects training data clusters based on their evolving informativeness.

Step 1: Difficulty-Aware Cluster Selection. Initially, we set the weights $W = \{w_1, w_2, ..., w_K\}$ for all clusters categorized by difficulty level, where w_i denotes the weight of cluster C_i and K is the number of clusters. To assess the difficulty score of each cluster, we employ the EXP3 [3] algorithm for the cluster selection. Specifically, for each iteration t, we first calculate the cluster score $DC_t(i)$ of the cluster C_i based on the cluster weight w_i, and then select a cluster (arm) DC_t^* with the highest score DC. The $DC_t(i)$ can be computed as:

$$DC_t(i) = (1-\gamma)\frac{w_i^{(t)}}{\sum_{j=1}^{K} w_j^{(t)}} + \frac{\gamma}{K} \tag{11}$$

where γ controls the exploration-exploitation trade-off.

The selected cluster at iteration t is the one with the highest probability: $C_{i^*} = \arg\max_{i \in [1,K]} DC_t(i)$.

Step 2: Sample Selection with IDU. After selecting a cluster C_i with the highest DC score, we apply our previously introduced

IDU utility function to sample the most informative subset B_{C_i} within the selected cluster C_i. Specifically, we select samples with the highest IDU scores to maximize utility gain at each iteration.

Step 3: Model Training and Reward Computation. Using the selected subset B_{C_i}, we train the large language model during iteration t. Once training is complete, we compute a reward $r_i^{(t)}$ to quantify the model's improvement resulting from the selected samples (Please refer to Section 5.3 for details).

Step 4: Cluster Weight Updates for Next Round Selection. After obtaining the reward $r_i^{(t)}$, we update the cluster weights $w_i^{(t+1)}$ according to EXP3 update rule:

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} \exp\left(\frac{\gamma}{K} \frac{r^{(t)}}{DC_t(i)}\right), & i = i_t \\ w_i^{(t)}, & \text{otherwise} \end{cases} \tag{12}$$

This adaptive weight-update mechanism ensures clusters that consistently yield high utility are progressively favored in subsequent iterations, achieving adaptive training data selection.

5.3 Self-Guided IDU-Based Reward

An effective reward function is critical to guiding effective cluster selection within the MAB framework. Ideally, such a reward should precisely capture each cluster's direct contribution to model improvement, while remaining computationally efficient and fully integrated into the training process.

Hence, we propose a *Self-Guided IDU-Based Reward*, leveraging our IDU utility to quantify each cluster's contribution to model improvement without additional inference overhead. Formally, the reward for training on cluster C_i at iteration t is computed as:

$$r_i^{(t)} = InfoGain(C_i, t) = \mathbb{E}_{x_i \in C_i}\left[IDU(\theta_{t-1}, x_i) - IDU(\theta_t, x_i)\right], \tag{13}$$

where θ_{t-1} and θ_t represent the model parameters before and after training, respectively. To maintain numerical stability and consistent scaling, rewards are further normalized to the range $[-1, 1]$ via min-max normalization.

Compared to traditional reward designs [9], our self-guided reward integrates into the standard training loop, accurately reflects dynamic model improvements at no additional inference cost, and significantly simplifies the reward computation.

6 THEORETICAL GUARANTEES

In this section, we analyze the theoretical guarantees of our IDU utility and the LEAD framework.

6.1 Optimal Smoothing Coefficient

We now analyze the optimal smoothing coefficient for the budget-constrained IDU optimization (Problem 1, presented in Section 5.1).

Theorem 6.1 (**Optimal Smoothing Coefficient**). *The optimal smoothing coefficient b^*:*

$$b^* = 1 - \frac{B}{n_0 T \cdot (1 + CV^2)} \tag{14}$$

where $n_0 = \alpha \cdot \overline{|C|}$ is the expected batch size without smoothing and heterogeneity effects.

Under a total budget B, we propose the optimization problem:

$$\max_{b,T} \sum_{t=1}^{T} \Delta IDU_t, \quad \text{s.t.} \sum_{t=1}^{T} n_t \le B \tag{15}$$

$$R^{(t)} = \Delta IDU_t = -(1-b)\eta_t |S_t| \Psi_t. \tag{16}$$

The specific simplification process can be referred to as Lemma 6.2.

Step 1: Estimate sample size selected in the t-th round n_t. $\mathbb{E}[n_t]$ can be simplified as follows (see Lemma 6.3 for details):

$$\mathbb{E}[n_t] = \alpha \cdot (1-b) \cdot \frac{\sum_{i=1}^{K} |C_i|^2}{\sum_{i=1}^{K} |C_i|} \cdot (1 + O(\gamma)) \tag{17}$$

Step 2: Estimate the expectation of utility gain ΔIDU_t. According to the Eq. (16) and Eq. (17), we can further obtain $\mathbb{E}[\Delta IDU_t]$.

$$\sum_{t=1}^{T} \mathbb{E}[\Delta IDU_t] = -n_0 \cdot (1-b)^2 \cdot (1 + \mathrm{CV}^2) \cdot \sum_{t=1}^{T} \eta_t \delta_t \tag{18}$$

Step 3: Redefine objective and constrained condition.

$$\max_{b,T} \sum_{t=1}^{T} \mathbb{E}[\Delta IDU_t], \quad \text{s.t.} \sum_{t=1}^{T} \mathbb{E}[n_t] \le B \tag{19}$$

$$\textit{where} \quad \mathbb{E}[n_t] = \alpha \cdot (1-b) \cdot |\overline{C}| \cdot (1 + \mathrm{CV}^2) \cdot (1 + O(\gamma)) \tag{20}$$

Let $\bar{\eta}\delta = \frac{1}{T}\sum_{t=1}^{T} \eta_t \delta_t$, The budget constraint becomes:

$$\sum_{t=1}^{T} \mathbb{E}[n_t] = \sum_{t=1}^{T} n_0 \cdot (1-b) \cdot (1 + \mathrm{CV}^2) \le B \tag{21}$$

Step 4: Solving optimal b^* and T^*. We formulate the Lagrangian:

$$\mathcal{L}(b, \lambda) = \mathbb{E}[\Delta IDU_t] - \lambda(\mathbb{E}[n_t] - B) \tag{22}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow 2\bar{\eta}\delta \cdot (1-b) = \lambda \tag{23}$$

We require $0 \le b^* < 1$, which implies:

$$T_{\min} = \left\lceil \frac{B}{n_0 \cdot (1 + \mathrm{CV}^2)} \right\rceil + 1 \tag{24}$$

Lemma 6.2 (**Batch Utility Change Decomposition**). *The utility change for batch S_t under the smoothed utility function can be expressed as:*

$$\Delta IDU_t = \begin{cases} -(1-b)\eta_t |S_t| \Psi_t + b|S_t|\delta_{t-1}(1 - b^{t-1}), & t \le 5 \\ -(1-b)\eta_t |S_t| \Psi_t, & t > 5 \end{cases} \tag{25}$$

where Ψ_t denotes the gradient alignment term:

$$\Psi_t = \beta_t^2 \delta_{t_k} + (1-\beta_t)^2 \delta_{t-1} + 2\beta_t(1-\beta_t)\sqrt{\delta_{t_k}\delta_{t-1}} \cos\phi_t \tag{26}$$

Proof. For any $x \in S_t$, $\Delta IDU_t(x)$ can be decomposed as:

$$\Delta IDU_t(x) = (1-b)\Delta L(\theta_t, x) + b(1-b)\sum_{k=0}^{t-3} b^k \Delta L(\theta_{t-2-k}, x) + (1-b)b^{t-1} IDU(\theta_0, x) \tag{27}$$

When $t > 5$, the exponential decay term b^{t-1} becomes negligible:

$$\Delta IDU_t \approx -(1-b)\eta_t |S_t| \Psi_t \tag{28}$$

□

Lemma 6.3 (**Expected Sample Size Under MAB mechanism**). *In the MAB framework using EXP3 for cluster selection with smoothed utility, the expected sample size per round $\mathbb{E}[n_t]$ satisfies:*

$$\mathbb{E}[n_t] = \alpha \cdot (1-b) \cdot \overline{|C|} \cdot (1 + CV^2) \cdot (1 + O(\gamma)) \tag{29}$$

where α is the sampling rate, b is the smoothing coefficient, $|C_i|$ is the size of cluster i, and γ is the exploration rate in function 11.

Proof. The reward signal for selecting cluster i at time t is:

$$R_i^{(t)} = \Delta IDU_t \propto (1-b)|C_i| \tag{30}$$

From the weight update Eq. (11) and Eq. (12) in the MAB EXP3 algorithm. As the algorithm converges to steady state, the weights stabilize such that:

$$\frac{w_i^{(t)}}{\sum_{j=1}^K w_j^{(t)}} \propto \exp\left(\sum_{\tau=1}^{t-1} \frac{\gamma}{K} \frac{R_i^{(\tau)}}{p_i^{(\tau)}}\right) \tag{31}$$

$$p_i^{(t)} \approx \frac{(1-\gamma)(1-b)|C_i|}{\sum_{j=1}^K (1-b)|C_j|} + \frac{\gamma}{K} \approx \frac{(1-b)|C_i|}{\sum_{j=1}^K |C_j|} + O(\gamma) \tag{32}$$

The expected sample size in round t is:

$$\mathbb{E}[n_t] = \alpha \sum_{i=1}^K p_i^{(t)} |C_i| = \alpha(1-b) \frac{\sum_{i=1}^K |C_i|^2}{\sum_{j=1}^K |C_j|} + O(\gamma) \tag{33}$$

Since $\sum_{i=1}^K |C_i| = N$ (total dataset size), we can express this as:

$$\mathbb{E}[n_t] = \alpha \cdot (1-b) \cdot \frac{\sum_{i=1}^K |C_i|^2}{\sum_{i=1}^K |C_i|} \cdot (1 + O(\gamma)) \tag{34}$$

□

6.2 Loss Changes in Gradient-Based Approximation

Recap that we have introduced utility function Eq. (6) in Section 3, In this section, we try to approximate the loss reduction $\Delta L'(\theta_t, x)$.

Theorem 6.4 (**IU Change Approximation**). *For any sample set S_t, the average uncertainty change $\Delta L'(\theta_t, S_t)$ when model parameters update from θ_{t-1} to θ_t can be approximated as:*

$$\delta_t \equiv \Delta L'(\theta_t, S_t) \tag{35}$$

$$= -\eta \left[\beta^2 \delta_{t_k} + (1-\beta)^2 \delta_{t-1} + 2\beta(1-\beta)\sqrt{\delta_{t_k}\delta_{t-1}} \cos\phi\right] \tag{36}$$

where ϕ is the angle between parameter update directions $\Delta\theta_{t_k}$ and $\Delta\theta_{t-1}$, with $\cos\phi = \frac{\Delta\theta_{t_k}^\top \Delta\theta_{t-1}}{\|\Delta\theta_{t_k}\| \|\Delta\theta_{t-1}\|}$.

$$\beta^* = \frac{\delta_{t-1} - \sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi}{\delta_{t_k} + \delta_{t-1} - 2\sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi} \tag{37}$$

Step 1: Simplify the loss change.

$$L(\theta_t, x) \approx L(\theta_{t-1}, x) + \nabla L(\theta_{t-1}, x)^\top (\theta_t - \theta_{t-1}) \tag{38}$$

Averaging over all samples in S_t:

$$\delta_t = \Delta L'(\theta_t, S_t) = -\eta_t \|\nabla L(\theta_{t-1}, S_t)\|^2 \tag{39}$$

Step 2: Approximate the gradient.

$$\nabla L'(S_t, \theta_{t-1}) \equiv \beta \cdot \nabla L(S_{t_k}, \theta_{t_k-1}) + (1-\beta) \cdot \nabla L(S_{t-1}, \theta_{t-2}), \tag{40}$$

where t_k is the most recent step when C_k was previously selected, C_k is the cluster selected at step t.

Step 3: Solving optimal β^* to obtain final IU Change Approximation $\Delta L'(\theta_t, S_t)$.

$$J(\beta) = \|\nabla L_t - (\beta \nabla L_{t_k} + (1-\beta)\nabla L_{t-1})\|^2 \tag{41}$$

Setting $\frac{d\tilde{J}}{d\beta} = 0$ yields the optimal coefficient:

$$\beta^* = \frac{\delta_{t-1} - \sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi}{\delta_{t_k} + \delta_{t-1} - 2\sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi}. \tag{42}$$

The loss change is then approximated as:

$$\delta_t = -\eta\left[(\beta^*)^2 \delta_{t_k} + (1-\beta^*)^2 \delta_{t-1} + 2\beta^*(1-\beta^*)\sqrt{\delta_{t_k}\delta_{t-1}}\cos\phi\right]. \tag{43}$$

7 EXPERIMENTS

7.1 Experimental Setup

Data Pool. To simulate realistic and diverse training scenarios, we construct two large-scale and heterogeneous data pools corresponding to different modalities.

(1) Text Data Pool comprises approximately **600,000 samples**. Our dataset integrates multiple well-established public sources, including WizardLM (ShareGPT) [45], WizardLM (Alpaca) [45], UltraChat [21], Standard Alpaca [61], unnatural [28], Alpaca code [13], MATH [27], GSM8K [20]. We closely follow Tulu [64] to process these datasets. All methods will select data from this pool for LLMs' instruction tuning.

Benchmarks and Metrics. We evaluate our method on four representative tasks aligned with the multi-task training pool but drawn from distinct distributions, reflecting key LLM capabilities.

- `Code Generation.` We use `HumanEval` [16] to evaluate the code-writing capabilities of LLMs. Performance is measured via the widely adopted `pass@1, pass@5` and `pass@10` metric.
- `Math Reasoning.` We use `GSM8k` [20] to evaluate the mathematical abilities of models. We adopt an 8-shot setting and evaluate performance using the `exact match accuracy` metric.
- `Multi-task Knowledge and Reasoning.` We evaluate on `MMLU` [26], which consists of a range of multiple-choice academic questions. We report `accuracy` as the metric.
- `Cross-lingual Question Answering.` To assess multilingual understanding, we utilize the `TYDIQA` [19]. We report `F1 scores` for passage selection and answer span extraction tasks.

Baselines. We study several existing state-of-the-art methods as our baselines for data selection.

(1) Full Data: Train the model using the entire data pool.

(2) Random Selection [68]: Randomly selects training samples.

(3) Instruction-Following Difficulty (IFD) [37]: Selects samples based on a complexity metric measuring instruction-following difficulty.

(4) Perplexity (PPL) [36]: Prioritizes uncertain samples with high perplexity.

(5) K-Center-Greedy (KCG) [57]: Maximizes diversity by iteratively choosing the sample farthest from the current selection.

(6) SelectIT [42]: Selects samples via uncertainty-aware self-reflection during instruction tuning.

(7) Token Length (TL) [68]: Selects samples with the longest response.

(8) ZIP [71]: prompting a strong LLM to estimate and select samples based on quality, relevance, and complexity scores.

(9) DiverseEvol [66]: Iteratively selects the most diverse samples using a K-Center-based strategy to self-evolve model performance.

(10) MIG [17]: Selects samples by maximizing information gain in semantic space using a label graph to balance quality and diversity.

Implementation Details of LEAD. We evaluate LEAD using three foundational models (LLAMA-3.1-8B, Mistral-7B and Qwen2-7B) and utilize Low-Rank Adaption (LoRA) [29] for parameter-efficient fine-tuning. The maximum learning rate is set as 2×10^{-5} with a linear decay schedule, and the batch size is 8. We also fix the maximum input sequence length to 3080. Models are trained for 4 epochs on 4 H800 GPUs. For the MAB setting, the number of arms is set to 7. The maximum sampling budget of LEAD is $15K$.

7.2 Exp-1: Overall Performance

We first evaluate LEAD and all baseline methods using the same budget of $15K$ samples, corresponding to 2.5% of the data pool.

Table 1 summarizes the evaluation results across four benchmarks and model architectures (LLaMA3.1-8B, Mistral-7B, and Qwen2-7B). Overall, LEAD consistently outperforms state-of-the-art baselines on most benchmarks, demonstrating its effectiveness.

(1) Consistent Effectiveness of LEAD across LLMs. LEAD consistently improves performance across LLMs: on LLaMA3.1-8B it reaches 66.62 (+6.31 over full data), with similar gains on Mistral-7B (+10.75) and Qwen2-7B (+6.09), confirming its robustness across architectures. Interestingly, we observe that perplexity-based selection performs well on GSM8K for stronger models like LLaMA3.1-8B and Qwen2-7B, but degrades on Mistral. This is likely due to its tendency to sample from uncertain regions of the data pool: strong models may already be confident on general-domain tasks and thus focus on math-relevant instructions, while weaker models like Mistral exhibit uncertainty across all domains, leading to task conflicts and catastrophic forgetting. LEAD avoids this issue by balancing instruction difficulty and task diversity, resulting in more stable gains across models.

(2) 2.5% of Data is All You Need. LEAD achieves these gains using only 2.5% of data, challenging the conventional assumption that larger datasets produce superior results. Specifically, our method outperforms full dataset training (`Full Data` baseline) across all model and benchmark settings. For example, On TYDIQA, it improves by 22.33, 29.15, and 12.63 points across the three models, respectively, demonstrating that selected instruction samples can lead to more effective learning.

(3) Outperforming State-of-the-art Baselines. LEAD outperforms both static and iterative selection methods with consistent effectiveness across models and benchmarks. While certain static methods demonstrate competitive performance in isolated settings (*e.g.,* SelectIT on LLaMA3.1-8B and PPL on Qwen2-7B), their effectiveness is often inconsistent across beckbones. Iterative methods generally achieve more stable results, yet LEAD attains the highest average performance. Although random sampling appears competitive in some cases, it suffers from high performance variance across runs.

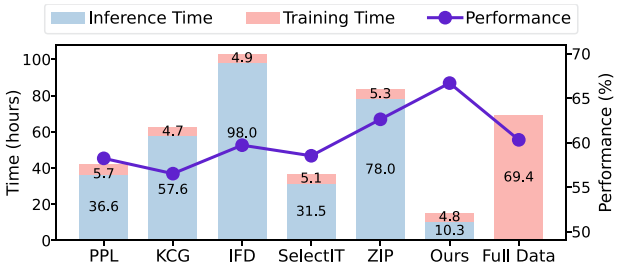

Figure 5: Inference time (Full Data) and training time (Selected Data) per iteration across different methods.

In contrast, our approach maintains consistent high performance across the benchmarks.

7.3 Exp-2: The Efficiency of LEAD

We evaluate the efficiency of LEAD compared to baseline methods (PPL, KCG, IFD, SelectIT, and ZIP) across four benchmarks. Note that we exclude `Random` and `TL` from this comparison, as these methods incur minimal computational overhead and were shown to perform significantly worse in **Exp-1**. We report the overall latency of all methods with *one round of selection iteration* on average.

Exp-2.1: Performance *vs.* Latency. We compare performance and inference latency (in log_2 scale) across different methods. As shown in Figure 8, LEAD (marked with a star) consistently achieves the best performance-latency trade-off, occupying the upper-left region of each plot. LEAD delivers a roughly 5× faster inference time compared to baselines, while maintaining top performance on benchmarks like TYDIQA, GSM8K, and HumanEval.

Exp-2.2: Analysis of Latency Composition. Figure 5 compares latency components (inference and training) of different methods. Inference time constitutes the computational bottleneck for traditional methods (*e.g.,* IFD: 98.0 hours), due to repeated full-dataset inference at each selection iteration. In contrast, LEAD requires inference only once (10.3 hours) for initial selection, eliminating subsequent inference overhead via inference-free IDU estimation.

7.4 Exp-3: Static *vs.* Iterative Data Selection

These experiments validate the necessity of iterative data selection.

Exp-3.1: Dynamics of Sample Utility over Training. We first track the overlap of samples initially identified as valuable (iteration 0) with the top-k samples in later iterations (1, 4, 7, and 10). As illustrated in Figure 6, the coverage rate for k=15,000 increases initially (from 0.77 to 0.98 at iteration 4), but significantly declines (to 0.67) in later iterations. These results underscore the dynamic nature of sample utility and the necessity of adapting data selection to the model's evolving state.

Exp-3.2: Performance of Static and Iterative Selection. We further compare the performance between one-round (static) and iterative selection strategies (Table 2). Iterative LEAD (IU) consistently surpasses One-round LEAD (IU), achieving an average improvement of 1.17 points (64.33 *vs.* 63.16). This performance gap confirms that iterative data selection is essential, as the utility of training samples dynamically changes throughout model training.

Table 1: Comparison of performance across different benchmarks with static data selection methods and iterative methods. indicates improvements over the second-best baseline.

Benchmark (Metric)	Static Data Selection Methods								Iterative Methods		
	Full Data	Random	PPL	TL	IFD	SelectIT	ZIP	MIG	KCG	DiverseEvol	LEAD (Ours)
LLaMA3.1-8B											
MMLU (Acc)	65.13	64.30	63.27	64.10	64.48	64.93	63.45	64.02	61.39	64.78	**65.40**
TYDIQA (F1)	50.94	40.91	41.89	46.47	55.66	61.33	45.41	49.93	43.12	48.84	**63.24**
GSM8K (EM)	56.63	54.80	56.32	54.28	43.52	54.89	57.32	53.76	51.73	53.96	**60.88**
HumanEval (Pass@10)	68.52	70.24	71.44	73.99	70.40	69.33	67.68	70.02	69.80	73.45	**76.95**
Average	56.41	55.09	49.95	56.80	56.10	57.86	54.84	55.86	53.74	57.04	**61.69 (+3.83)**
Mistral-7B											
MMLU (Acc)	61.45	61.68	62.38	61.93	61.65	64.93	61.93	61.47	61.02	61.32	**62.10**
TYDIQA (F1)	49.63	38.02	52.72	39.88	41.41	36.79	42.04	40.02	39.79	42.19	**67.17**
GSM8K (EM)	40.56	33.51	22.82	37.76	31.77	35.86	41.17	35.71	33.89	34.46	**45.26**
HumanEval (Pass@10)	58.37	57.35	54.68	60.54	52.05	58.15	61.91	57.12	59.96	55.17	**59.01**
Average	47.04	43.34	43.87	46.46	43.34	45.60	47.67	45.01	45.14	43.99	**51.73 (+4.06)**
Qwen2-7B											
MMLU (Acc)	70.54	69.85	70.70	70.52	70.03	70.32	70.54	70.26	70.64	70.36	**70.19**
TYDIQA (F1)	42.94	43.43	42.63	38.91	35.00	43.80	34.51	40.92	40.92	39.61	**56.06**
GSM8K (EM)	73.16	73.16	79.00	78.53	74.91	74.60	75.66	76.18	76.04	78.57	**79.83**
HumanEval (Pass@10)	82.56	79.51	78.44	80.79	81.94	78.14	83.91	78.81	78.81	79.73	**84.22**
Average	62.20	61.69	62.37	62.34	61.40	62.05	61.68	61.66	61.93	62.21	**66.38 (+4.01)**

Table 2: Comparison between IU and IDU. LEAD (IDU) refers to LEAD using IDU as the utility function. One-round and Iterative LEAD (IU) denote non-iterative and iterative variants of the IU approach.

Method	Benchmarks				Average
	MMLU	TYDIQA	GSM8K	HumanEval	
LEAD (IDU)	**65.40**	**63.24**	**60.88**	**76.95**	**66.62**
One-round LEAD (IU)	63.92	59.13	57.47	72.13	63.16
Iterative LEAD (IU)	64.72	60.15	57.99	74.46	64.33

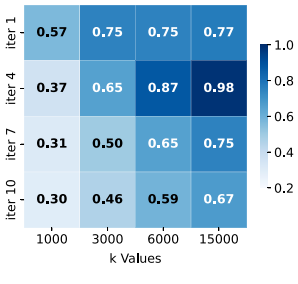

Figure 6: Coverage of Top-k samples between iter. t and iter. 0.

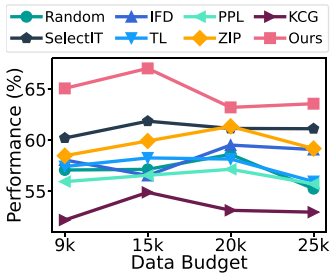

Figure 7: Avg performance by varying training budgets.

7.5 Exp-4: Ablation Study of LEAD

Exp-4.1: Ablation Study on LEAD Components. To validate the effectiveness of our proposed framework, we conduct an ablation study on the LLaMA3.1-8B model and Qwen2-7B by systematically removing individual modules of our LEAD framework.

As shown in Table 3, removing any module leads to a performance drop: average metric decreases by 1.78 (MAB), 1.23 (TC), and 3.27 (IDU). The IDU module has the most pronounced impact, particularly on TYDIQA (-7.36), underscoring its role in identifying informative samples. Removing the TC module also degrades performance across all benchmarks, confirming the value of semantic

Table 3: Ablation study of different modules on LLaMA3.1-8B and Qwen2-7B.

Models	Module			Benchmarks				Average
	MAB	TC	IDU	MMLU	TYDIQA	GSM8K	HumanEval	
LLaMA3.1-8B	✓			64.83	59.84	54.81	72.13	62.90
		✓		62.71	61.31	51.48	74.25	62.44
			✓	64.13	61.47	57.92	74.93	64.61
	✓	✓		65.10	55.88	57.99	74.41	63.35
	✓		✓	64.70	66.46	55.95	74.46	65.39
		✓	✓	65.30	64.29	56.40	73.38	64.84
	✓	✓	✓	**65.40**	**63.24**	**60.88**	**76.95**	**66.62**
Qwen2-7B	✓			69.17	42.61	78.09	75.10	66.24
		✓		69.87	39.17	77.10	73.05	64.80
			✓	70.05	43.98	79.00	79.16	68.05
	✓	✓		70.01	45.85	77.41	72.89	66.54
	✓		✓	70.54	50.16	79.21	80.07	69.89
		✓	✓	70.16	48.14	78.43	77.13	68.47
	✓	✓	✓	**70.19**	**56.06**	**79.83**	**84.22**	**72.58**

Table 4: Ablation study of LEAD framework.

Method	Replace Strategy	Benchmarks				Average
		MMLU	TYDIQA	GSM8K	HumanEval	
Reward Function	IFD-MAB	65.29	65.31	51.02	72.13	63.44
	PPL-MAB	65.52	67.17	51.71	72.11	64.13
IDU	Random	65.10	55.88	57.99	74.41	63.35
	PPL	64.13	49.40	52.53	68.17	59.59
	IU	64.72	60.15	57.99	74.46	63.56
	IFD	64.92	54.98	51.86	70.71	60.62
MAB	Random	64.17	61.46	55.95	74.00	63.90
	Easy2Hard	64.73	60.32	58.98	71.81	63.96
	Hard2Easy	64.29	61.96	56.65	74.54	64.36
TC	General-Purpose	64.82	62.14	56.28	74.25	64.37
Ours	-	**65.40**	**63.24**	**60.88**	**76.95**	**66.62**

clustering. The removal of the MAB module significantly affects performance on the challenging GSM8K (-4.48), demonstrating its role in balancing exploration and exploitation. For Qwen2-7B, the same trend holds: removing IDU yields the largest drop (-4.52), followed by MAB (-2.34) and TC (-1.92). Overall, the ablation study highlights the effectiveness of each component within the LEAD framework.

Exp-4.2: The Effectiveness of IDU Utility. To demonstrate the effectiveness of IDU, we conducted comprehensive experiments examining its performance from two perspectives.

First, to verify that IDU effectively smooths the instability issues during iterative selection, we compared LEAD (IDU) against LEAD (IU) on LLaMA3.1-8B. As shown in Table 2, LEAD (IDU) consistently outperforms LEAD (IU) across benchmarks (+3.06%, 66.62 *vs.* 63.56), confirming its smoothing design effectively mitigates instability. Second, to validate IDU's superiority as a utility function, we compared it against alternative utility metrics while keeping other LEAD components intact. The results in Table 4 show that replacing IDU with conventional metrics like PPL leads to dramatic performance degradation (from 66.62 to 59.59). These findings highlight IDU's robustness as a reliable criterion for selecting high-value samples across diverse tasks.

Exp-4.3: The Effectiveness of MAB Module. To assess the MAB module's contribution, we compare it against three baselines: (1) Random-LEAD: random selection of difficulty-aware clusters per iteration; (2) Easy2Hard-LEAD: iterative training from easy to hard clusters based on difficulty scores; and (3) Hard2Easy-LEAD: iterative training from hard to easy. For a fair comparison, all modules except the training strategy remained consistent with the LEAD.

As shown in Table 4, our MAB training schedule significantly outperforms the other three strategies, confirming its effectiveness in dynamically balancing exploration and exploitation. By adaptively selecting difficulty-aware clusters, MAB enhances both overall performance and generalizability. In contrast, Easy2Hard-LEAD yields the low score (63.96), highlighting the limitations of traditional curriculum learning in instruction tuning, as a fixed progression from easy to hard can hinder learning dynamics and lead to premature convergence. Hard2Easy-LEAD performs slightly better (64.36), yet still underperforms compared to MAB, indicating that prioritizing difficult clusters does not guarantee optimal results.

Exp-4.4: The Effectiveness of Reward Function. We assess the effectiveness of our proposed IDU-based reward by comparing it with two widely-used reward metrics: IFD [37] and PPL [36]. As shown in Table 4, our IDU-based reward consistently achieves the best overall performance (average 66.62), surpassing IFD (63.44) and PPL (64.13). This demonstrates that directly measuring the reduction in instance-level dynamic uncertainty provides more effective guidance for cluster selection than traditional metrics.

Exp-4.5: The Effectiveness of Task-Specific Clustering. We evaluate task-specific clustering via two ablations: (1) removing the clustering module, and (2) replacing task-specific with general-purpose embeddings.

The results presented in Table 3 and Table 4 show that module removal causes greater degradation than embedding replacement, underscoring its key role in organizing data by task relevance. While general-purpose embeddings capture broad semantics, they fail to represent task-specific nuances, yielding less effective clustering.

7.6 Exp-5: Effect of Sample Size on Performance

To examine the impact of data selection strategies on data budgets' effectiveness, we conduct experiments using subsets with varying budgets. As illustrated in Figure 7, LEAD consistently presents higher average performance than alternative selection methods across all data budgets, achieving peak performance with only 15K samples. Even the second-best method with a 25K sample budget still underperforms LEAD at 15K, highlighting the superior sample efficiency and effectiveness of our approach. Notably, we observe a non-linear performance curve: gains taper and eventually decline beyond a certain data threshold, which reveals a crucial insight: "alignment-suitable data" is inherently limited. This finding challenges the conventional wisdom that more data automatically yields better results, underscoring the critical importance of strategic data selection over mere quantity.

7.7 Exp-6: Parameter Sensitivity Analysis

Exp-6.1: Effect of Sampling Threshold α. As shown in Figure 9, performance peaks when α is between 0.15 and 0.20, reaching a balance between iteration quantity and quality. Higher α values yield more samples per round but fewer iterations, limiting adaptability. Lower values allow more iterations but provide weaker signals.

Exp-6.2: Effect of Smoothing Coefficient b of IDU. Figure 10(a) shows optimal performance at b=0.1, which effectively balances stability and responsiveness. Smaller values (b < 0.1) overemphasize current fluctuations, leading to noise susceptibility, whereas larger values (b > 0.2) overweight historical signals, reducing adaptability.

Exp-6.3: Effect of Exploration Rate γ of MAB. Figure 10(b) shows that our MAB algorithm achieves optimal performance at moderate exploration rates (γ=0.05-0.07). Minimal exploration (γ=0.01) limits discovery of new clusters, whereas excessive exploration (γ=0.12) hinders focus on promising clusters.

Exp-6.4: Effect of Different Clustering Algorithm of LEAD. We compared Agglomerative Clustering, DBSCAN, and K-Means. The results in Figure 10 (c) show minimal differences (66.42–67.02), suggesting that LEAD is not sensitive to the choice of clustering algorithm and is robust across methods.

Exp-6.5: Effect of the Number of Clusters k (MAB Arms). In this experiment, we evaluate the impact of varying the number of clusters (k) on LEAD's performance, where k also represents the number of arms in the MAB algorithm. The results presented in Figure 10 (d) show that the performance fluctuates as k changes. The best average performance (67.02) is observed when k = 7. As k increases further, the performance begins to decline, with k = 15 achieving the lowest average score of 60.03. This indicates that a moderate number of clusters (arms) provides the best balance between selection diversity and efficiency.

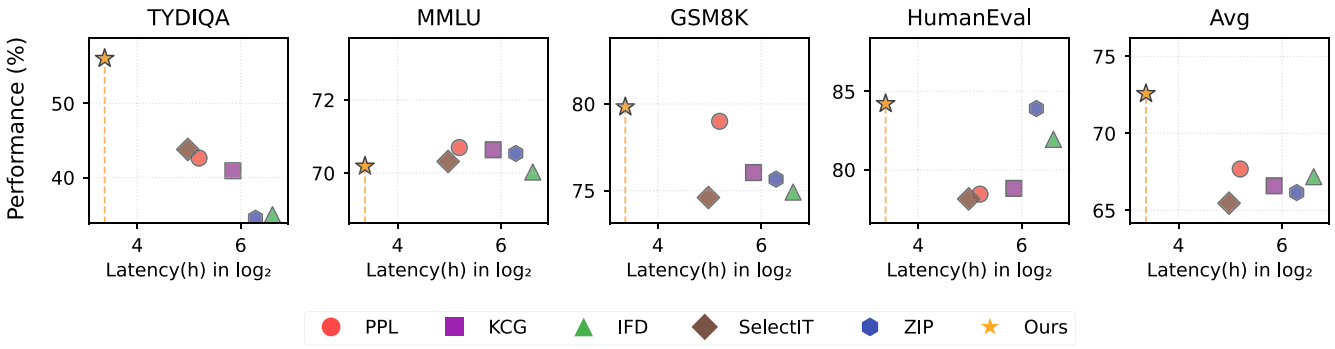

Figure 8: Comparison of Performance(y-axis) and Latency(x-axis) across six data selection methods.

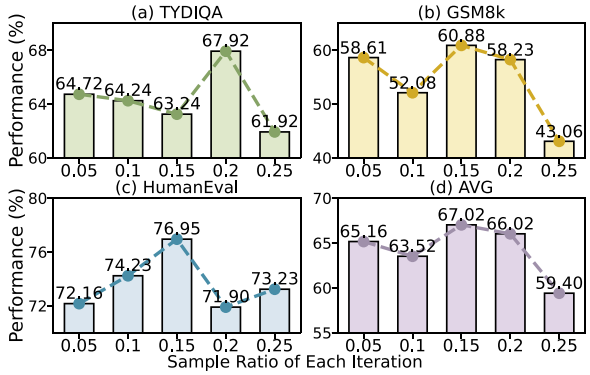

Figure 9: Performance on sample ratios of each iteration.

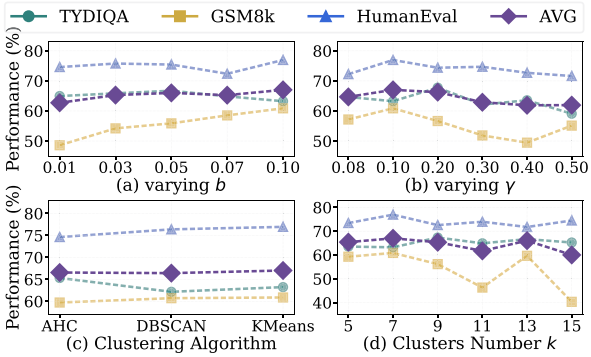

Figure 10: Parameter sensitivity analysis.

8 RELATED WORK

Data Selection for Instruction Tuning. Previous works on data selection [10, 25, 67, 74] can be broadly categorized into two key approaches: model-agnostic methods and model-aware methods.

Model-agnostic methods operate independently of the target model, including rule-based approaches [5, 6, 30, 45, 54, 58, 76] that are computationally efficient but lack semantic understanding. Advanced model-based methods [14, 15, 40] like GPT-4 [1] that provide nuanced assessment at high computational cost, and proxy model-based methods [36, 70] that balance efficiency and quality. However, these methods cannot adapt to the specific learning characteristics of the target model. Model-aware methods [5, 8, 9, 41, 46, 47, 73] address this limitation by customizing selection based on the model's learning dynamics, though they introduce higher computational costs through required model inference or fine-tuning. In contrast, LEAD proposes a two-stage adaptive approach that efficiently combines model-aware adaptiveness with zero computational overhead, effectively addressing the challenge of balancing effectiveness and efficiency in instruction tuning data selection.

Sample Utility Estimation. Sample utility scoring is central to data selection, leveraging diverse metrics [8, 56, 65]. Perplexity-based metrics [36, 53] prefer simple patterns, whereas diversity-aware strategies [66, 72] broaden coverage but always hinge on embedding quality. Quality-oriented metrics, such as influence scoring [18, 23, 31, 67], external model [38] and gradient matching [7, 8], are principled but gradient-expensive. Complexity-based selection [37, 43] risks noisy samples, while uncertainty-driven metrics [24, 42] suffer from loss landscape instability. Recent advances like Quad [73] enhance efficiency via MAB-driven utility estimation, yet still depend on extra inference and lack difficulty-aware adaptation. In contrast, we introduce IDU, an inference-free utility function that achieves zero-cost estimation while preserving selection effectiveness.

9 CONCLUSION

In this paper, we proposed LEAD, an iterative data selection framework for LLMs instruction tuning. LEAD introduces an Instance-Level Dynamic Uncertainty utility function, enabling accurate utility estimation without extra inference. In addition, we developed a coarse-to-fine selection approach guided by a multi-armed bandit mechanism. Experiments show LEAD achieves 6.1%-10.8% performance improvement using only 2.5% training data and reduces training costs by 5-10×.

ACKNOWLEDGMENTS

This paper was supported by National Key R&D Program of China (2024YFA1012700); the NSF of China (62402409); EU Horizon project DataGEMS (101188416); Youth S&T Talent Support Programme of Guangdong Provincial Association for Science and Technology (SKXRC2025461); the Young Talent Support Project of Guangzhou Association for Science and Technology (QT-2025-001); Guangzhou-HKUST(GZ) Joint Funding Program (2025A03J3714); Guangzhou Basic and Applied Basic Research Foundation (2025A04J3935); and Guangdong Provincial Project (2023CX10X008).

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. [n. d.]. A Survey on Data Selection for Language Models. *Transactions on Machine Learning Research* ([n. d.]).

[3] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32, 1 (2002), 48–77.

[4] Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. 2024. Data Diversity Matters for Robust Instruction Tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 3411–3425.

[5] Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: Instruction data selection for tuning large language models. *arXiv preprint arXiv:2307.06290* (2023).

[6] Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. 2020. Human-in-the-loop Outlier Detection. In *SIGMOD Conference*. ACM, 19–33.

[7] Chengliang Chai, Kaisen Jin, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, Guoliang Li, Ye Yuan, and Guoren Wang. 2025. Cost-effective Missing Value Imputation for Data-effective Machine Learning. *ACM Transactions on Database Systems* 50, 3 (2025), 1–36.

[8] Chengliang Chai, Jiabin Liu, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, and Guoliang Li. 2023. Goodcore: Data-effective and data-efficient machine learning through coreset selection over incomplete data. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.

[9] Chengliang Chai, Jiabin Liu, Nan Tang, Guoliang Li, and Yuyu Luo. 2022. Selective data acquisition in the wild for model charging. *Proceedings of the VLDB Endowment* 15, 7 (2022), 1466–1478.

[10] Chengliang Chai, Nan Tang, Ju Fan, and Yuyu Luo. 2023. Demystifying Artificial Intelligence for Data Preparation. In *SIGMOD Conference Companion*. ACM, 13–20.

[11] Chengliang Chai, Jiayi Wang, Yuyu Luo, Zeping Niu, and Guoliang Li. 2023. Data Management for Machine Learning: A Survey. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 4646–4667.

[12] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.

[13] Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation.

[14] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. [n. d.]. AlpaGasus: Training a Better Alpaca with Fewer Data. In *The Twelfth International Conference on Learning Representations*.

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

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

[17] Yicheng Chen, Yining Li, Kai Hu, Zerun Ma, Haochen Ye, and Kai Chen. 2025. Mig: Automatic data selection for instruction tuning by maximizing information gain in semantic space. *arXiv preprint arXiv:2504.13835* (2025).

[18] Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. 2024. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954* (2024).

[19] Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. *Transactions of the Association for Computational Linguistics* 8 (2020), 454–470.

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

[21] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 3029–3051.

[22] Everette S Gardner Jr. 1985. Exponential smoothing: The state of the art. *Journal of forecasting* 4, 1 (1985), 1–28.

[23] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*. PMLR, 2242–2251.

[24] Jindong Han, Hao Liu, Jun Fang, Naiqiang Tan, and Hui Xiong. [n. d.]. Automatic Instruction Data Selection for Large Language Models via Uncertainty-Aware Influence Maximization. In *THE WEB CONFERENCE 2025*.

[25] LIU Hanmo, DI Shimin, LI Haoyang, LI Shuangyin, CHEN Lei, and ZHOU Xiaofang. 2024. Effective Data Selection and Replay for Unsupervised Continual Learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1449–1463.

[26] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. [n. d.]. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

[27] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874* (2021).

[28] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689* (2022).

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

[30] Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems* 36 (2023), 47669–47681.

[31] Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. [n. d.]. DataInf: Efficiently Estimating Data Influence in LoRA-tuned LLMs and Diffusion Models. In *The Twelfth International Conference on Learning Representations*.

[32] Boyan Li, Chong Chen, Zhujun Xue, Yinan Mei, and Yuyu Luo. 2025. DeepEye-SQL: A Software-Engineering-Inspired Text-to-SQL Framework. *CoRR* abs/2510.17586 (2025).

[33] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024. The Dawn of Natural Language to SQL: Are We Fully Ready? [Experiment, Analysis & Benchmark]. *Proc. VLDB Endow.* 17, 11 (2024), 3318–3331.

[34] Boyan Li, Jiayi Zhang, Ju Fan, Yanwei Xu, Chong Chen, Nan Tang, and Yuyu Luo. 2025. Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search. In *ICML*. OpenReview.net.

[35] Changlun Li, Chenyu Yang, Yuyu Luo, Ju Fan, and Nan Tang. 2025. Weak-to-Strong Prompts with Lightweight-to-Powerful LLMs for High-Accuracy, Low-Cost, and Explainable Data Transformation. *Proc. VLDB Endow.* 18, 8 (2025), 2371–2384.

[36] Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024. Superfiltering: Weak-to-Strong Data Filtering for Fast Instruction-Tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 14255–14273.

[37] Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024. From Quantity to Quality: Boosting LLM Performance with Self-Guided Data Selection for Instruction Tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 7595–7628.

[38] Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, et al. 2024. One-Shot Learning as Instruction Data Prospector for Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 4586–4601.

[39] Yiwei Li, Jiayi Shi, Shaoxiong Feng, Peiwen Yuan, Xinglin Wang, Boyuan Pan, Heda Wang, and Yao Hu. 2024. Instruction Embedding: Latent Representations of Instructions Towards Task Identification. *Advances in Neural Information Processing Systems* 37 (2024), 87683–87711.

[40] W Lian et al. 2023. SlimOrca: An Open Dataset of GPT-4 Augmented FLAN Reasoning Traces, with Verification.

[41] Jiabin Liu, Chengliang Chai, Yuyu Luo, Yin Lou, Jianhua Feng, and Nan Tang. 2022. Feature augmentation with reinforcement learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 3360–3372.

[42] Liangxin Liu, Xuebo Liu, Derek F Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024. Selectit: Selective instruction tuning for large language models via uncertainty-aware self-reflection. *arXiv preprint arXiv:2402.16705* (2024).

[43] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024. What Makes Good Data for Alignment? A Comprehensive Study of Automatic Data Selection in Instruction Tuning. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=BTKAeLqLMw

[44] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going? *IEEE Trans. Knowl. Data Eng.* 37, 10 (2025), 5735–5754.

[45] Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. [n. d.]. # InsTag: Instruction Tagging for Analyzing Supervised Fine-tuning of Large Language Models. In *The Twelfth International Conference on Learning Representations*.

[46] Yuyu Luo, Chengliang Chai, Xuedi Qin, Nan Tang, and Guoliang Li. 2020. Interactive Cleaning for Progressive Visualization through Composite Questions. In *ICDE*. IEEE, 733–744.

[47] Yuyu Luo, Chengliang Chai, Xuedi Qin, Nan Tang, and Guoliang Li. 2020. VisClean: Interactive Cleaning for Progressive Visualization. *Proc. VLDB Endow.* 13, 12 (2020), 2821–2824.

[48] Yuyu Luo, Guoliang Li, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Natural Language to SQL: State of the Art and Open Problems. *Proc. VLDB Endow.* 18, 12 (2025), 5466–5471.

[49] Yuyu Luo, Xuedi Qin, Chengliang Chai, Nan Tang, Guoliang Li, and Wenbo Li. 2022. Steerable Self-Driving Data Visualization. *IEEE Trans. Knowl. Data Eng.* 34, 1 (2022), 475–490.

[50] Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. 2018. DeepEye: Towards Automatic Data Visualization. In *ICDE*. IEEE Computer Society, 101–112.

[51] Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021. Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks. In *SIGMOD Conference*. ACM, 1235–1247.

[52] Yuyu Luo, Yihui Zhou, Nan Tang, Guoliang Li, Chengliang Chai, and Leixian Shen. 2023. Learned Data-aware Image Representations of Line Charts for Similarity Search. *Proc. ACM Manag. Data* 1, 1 (2023), 88:1–88:29.

[53] Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564* (2023).

[54] Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. 2023. Octopack: Instruction tuning code large language models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

[55] Xuedi Qin, Yuyu Luo, Nan Tang, and Guoliang Li. 2020. Making data visualization more efficient and effective: a survey. *VLDB J.* 29, 1 (2020), 93–117.

[56] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB endowment. International conference on very large data bases*, Vol. 11. 269.

[57] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

[58] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15725–15788.

[59] Jielin Song, Siyu Liu, Bin Zhu, and Yanghui Rao. 2024. IterSelectTune: An Iterative Training Framework for Efficient Instruction-Tuning Data Selection. *arXiv preprint arXiv:2410.13464* (2024).

[60] Wangtao Sun, Haotian Xu, Xuanqing Yu, Pei Chen, Shizhu He, Jun Zhao, and Kang Liu. 2024. ItD: Large Language Models Can Teach Themselves Induction through Deduction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2719–2731.

[61] 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.

[62] Joannes Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*. Springer, 437–448.

[63] Jiachen Tianhao Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. 2024. GREATS: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems* 37 (2024), 131197–131223.

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

[65] Yong Wang, Kaiyu Li, Yuyu Luo, Guoliang Li, Yunyan Guo, and Zhuo Wang. 2024. Fast, Robust and Interpretable Participant Contribution Estimation for Federated Learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2298–2311.

[66] Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023. Self-evolved diverse data sampling for efficient instruction tuning. *arXiv preprint arXiv:2311.08182* (2023).

[67] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333* (2024).

[68] Tingyu Xia, Bowen Yu, Kai Dang, An Yang, Yuan Wu, Yuan Tian, Yi Chang, and Junyang Lin. 2024. Rethinking data selection at scale: Random selection is almost all you need. *arXiv preprint arXiv:2410.09335* (2024).

[69] Chenyu Yang, Yuyu Luo, Chuanxuan Cui, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Data Imputation with Limited Data Redundancy Using Data Lakes. *Proc. VLDB Endow.* 18, 10 (2025), 3354–3367.

[70] Yu Yang, Siddhartha Mishra, Jeffrey Chiang, and Baharan Mirzasoleiman. 2024. Smalltolarge (s2l): Scalable data selection for fine-tuning large language models by summarizing training trajectories of small models. *Advances in Neural Information Processing Systems* 37 (2024), 83465–83496.

[71] Mingjia Yin, Chuhan Wu, Yufei Wang, Hao Wang, Wei Guo, Yasheng Wang, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. 2024. Entropy law: The story behind data compression and llm performance. *arXiv preprint arXiv:2407.06645* (2024).

[72] Simon Yu, Liangyu Chen, Sara Ahmadian, and Marzieh Fadaee. 2024. Diversify and Conquer: Diversity-Centric Data Selection with Iterative Refinement. *arXiv preprint arXiv:2409.11378* (2024).

[73] Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ju Fan, et al. 2024. Harnessing Diversity for Important Data Selection in Pretraining Large Language Models. *arXiv preprint arXiv:2409.16986* (2024).

[74] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems* 36 (2023), 55006–55021.

[75] Yizhang Zhu, Liangwei Wang, Chenyu Yang, Xiaotian Lin, Boyan Li, Wei Zhou, Xinyu Liu, Zhangyang Peng, Tianqi Luo, Yu Li, Chengliang Chai, Chong Chen, Shimin Di, Ju Fan, Ji Sun, Nan Tang, Fugee Tsung, Jiannan Wang, Chenglin Wu, Yanwei Xu, Shaolei Zhang, Yong Zhang, Xuanhe Zhou, Guoliang Li, and Yuyu Luo. 2025. A Survey of Data Agents: Emerging Paradigm or Overstated Hype? *CoRR* abs/2510.23587 (2025).

[76] Terry Yue Zhuo, Armel Zebaze, Nitchakarn Suppattarachai, Leandro von Werra, Harm de Vries, Qian Liu, and Niklas Muennighoff. 2024. Astraios: Parameter-efficient instruction tuning code large language models. *arXiv preprint arXiv:2401.00788* (2024).