ClusterUCB: Efficient Gradient-Based Data Selection for Targeted Fine-Tuning of LLMs

Zige Wang[1], Qi Zhu[2], Fei Mi, Minghui Xu[3], Ruochun Jin[4], Wenjing Yang[4*]

[1]School of Computer Science, Peking University,
[2] Tsinghua University, [3] Tianjin University,
[4]College of Computer Science and Technology, National University of Defense Technology

Correspondence: wenjing.yang@nudt.edu.cn

Abstract

Gradient-based data influence approximation has been leveraged to select useful data samples in the supervised fine-tuning of large language models. However, the computation of gradients throughout the fine-tuning process requires too many resources to be feasible in practice. In this paper, we propose an efficient gradient-based data selection framework with clustering and a modified Upper Confidence Bound (UCB) algorithm. Based on the intuition that data samples with similar gradient features will have similar influences, we first perform clustering on the training data pool. Then, we frame the inter-cluster data selection as a constrained computing budget allocation problem and consider it a multi-armed bandit problem. A modified UCB algorithm is leveraged to solve this problem. Specifically, during the iterative sampling process, historical data influence information is recorded to directly estimate the distributions of each cluster, and a cold start is adopted to balance exploration and exploitation. Experimental results on various benchmarks show that our proposed framework, ClusterUCB, can achieve comparable results to the original gradient-based data selection methods while greatly reducing computing consumption. The code implementation can be found at https://github.com/ZigeW/ClusterUCB.

1 Introduction

Data selection has been a challenging problem in the Supervised Fine-Tuning (SFT) of Large Language Models (LLMs) (Wang et al., 2023b; Albalak et al., 2024). Some researchers propose to use the data influence approximation (Hampel, 1974) to select data samples with the highest influence on target loss optimization during the training process (Charpiat et al., 2019; Pruthi et al., 2020; Xia et al., 2024; Wang et al., 2025). The data influence at a certain training step is approximated as the inner products or the cosine similarities of the gradients of the training and target validation data samples. Then, the one-step data influence approximations are computed after every short period and aggregated through the training process.

Although proven to be effective, the calculation of data influence approximation consumes many computing resources and can be infeasible in practice when the computing budget is restricted. To reduce resource consumption, one simple way is to compute the gradients of all data samples after longer training periods. Previous work (Wang et al., 2025) shows that the data influence approximated with gradients will soon lose its indication after multiple training steps. Hence, simply extending the interval of every two times of gradient computation is likely to result in inferior selected data subsets.

Another way to lower the computation cost is to reduce the number of data samples needed in the calculation of one-step data influence approximation while maintaining the ability to select the data samples with the highest influences. From the derivation of data influence approximation, we come to an intuition that the training data samples with similar gradients tend to have similar influences on the same target loss optimization. Hence, we first perform clustering on all training data samples at the beginning of training according to the similarities of their gradients. In this way, data samples with high influence tend to be concentrated into a few clusters. By picking out the clusters with higher probabilities to contain high-influence data samples, we can avoid the calculation over a large number of low-influence training data samples.

With a constrained computing budget, we frame the data selection among clusters as a computing budget allocation problem, which we call the inter-cluster data selection. However, a challenge lies in the unknown influence distribution of each cluster. To tackle this challenge, we consider it as a multi-armed bandit problem with each cluster as

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18867–18880
November 4-9, 2025

one arm, and the reward of drawing this arm is the influence of a randomly chosen sample (without replacement) from this cluster. Then, we adapt the most commonly used Upper Confidence Bound (UCB) algorithm to the inter-cluster data selection problem. To better suit our objective, we record the historical rewards and directly estimate the distribution of each arm. A cold start is added to improve the initial estimations. Combining clustering with the modified UCB algorithm, we propose an efficient gradient-based data selection framework **ClusterUCB** that can be applied to various gradient-based data selection methods. To verify the effectiveness of ClusterUCB, we evaluate it on four widely used benchmarks and two state-of-the-art gradient-based data selection methods (Xia et al., 2024; Wang et al., 2025). Experimental results demonstrate that ClusterUCB can achieve comparable results with the original gradient-based data selection methods while greatly reducing the computing consumption.

2 Related Work

SFT Data Selection of LLMs Selecting suitable data samples for the supervised fine-tuning of large language models has been an ongoing hot topic in the research community (Wang et al., 2023b; Albalak et al., 2024). Extended works are proposed to address different issues of SFT data selection, such as data quality (Zhou et al., 2023; Cao et al., 2023; Lu et al., 2023a), diversity (Lu et al., 2023b; Wan et al., 2023; Ding et al., 2023), complexity (He et al., 2024; Zhao et al., 2024), and so on. While some works try to improve multi-task SFT through proper task composition (Dong et al., 2024; Kung et al., 2023), there are also works dedicated to the data selection problem in targeted SFT focusing on a single task (Chen et al., 2024; Xia et al., 2024; Wang et al., 2025). In these works, gradient-based data influence approximation is used to evaluate the value of individual data samples in the process of targeted SFT. For example, *LESS* (Xia et al., 2024) adapts the classic data influence approximation to Adam optimizer and LoRA (Hu et al., 2022) training. *Dynamic* (Wang et al., 2025) points to the decreasing effectiveness of selection during the long-time training process and proposes dynamically updating the selected data coresets. Although effective, these proposed gradient-based data selection methods require many resources in practice. Lin et al. (2024) propose a cache-and-retrieve method to achieve memory-efficient gradient caching and fast data influence estimation of every training data sample. Different from them, our work addresses the computational efficiency and proposes a gradient-based data selection framework to reduce the number of training samples that require gradient computation during the data selection process.

Individual data influence Data influence function is proposed to evaluate the influence of data samples on model training (Hampel, 1974). Since the evaluation of different combinations of data samples is too costly, some researchers tend to evaluate the influence of individual data samples and treat the sum of these influences as the influence of a data subset. There are two branches in the individual data influence approximation: one is auxiliary model learning and simulation (Ilyas et al., 2022; Guu et al., 2023; Liu et al., 2024; Engstrom et al., 2024), and the other is gradient-based training dynamic approximation (Charpiat et al., 2019; Pruthi et al., 2020; Xia et al., 2024; Wang et al., 2025; Pan et al., 2025). In our work, we propose a framework to efficiently apply gradient-based individual data influence approximation in the SFT data selection of LLMs.

3 Methodology

Our work focuses on the efficient data selection for supervised fine-tuning of large language models on a target task. Given a pretrained model with parameter $\boldsymbol{\theta}$, the training loss $\mathcal{L}(\cdot;\boldsymbol{\theta})$, the training data pool $\mathbf{X}_{tr} = \{\mathbf{x}_{tr}^1, \mathbf{x}_{tr}^2, ..., \mathbf{x}_{tr}^N\}$, the target task T, and the validation data samples representing the target task $\mathbf{X}_v = \{\mathbf{x}_v^1, \mathbf{x}_v^2, ..., \mathbf{x}_v^M\}$, our goal is to select the training data subset with the highest data influence on the training process targeting task T under the constraint of limited computing budget.

3.1 Preliminary

Before elaborating on our proposed framework, we first introduce the calculation of gradient-based data influence approximation and the previously proposed data selection methods based on it.

3.1.1 Gradient-based data influence approximation

At time step t in the training process, the model parameter is $\boldsymbol{\theta}^t$. Considering the training data sample $\mathbf{x}_{tr}^i$ and validation data sample $\mathbf{x}_v^j$, the one-step influence $\mathcal{I}^t$ of $\mathbf{x}_{tr}^i$ is defined as the amount of

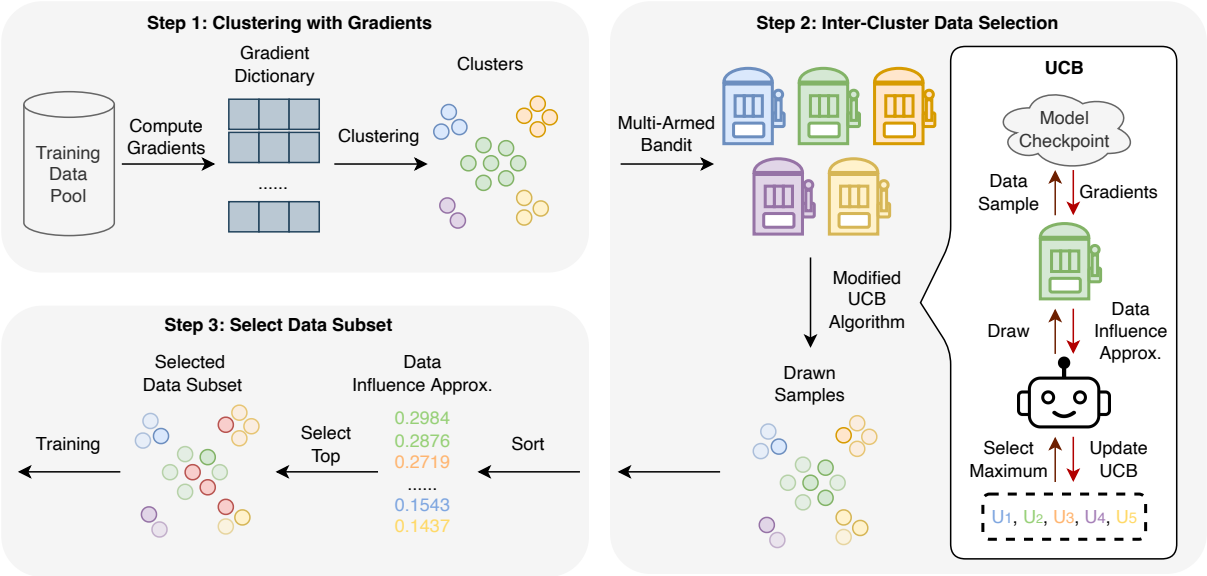


Figure 1: Illustration of ClusterUCB. Step 1: We first compute gradients of all training data samples and perform clustering according to their cosine similarities. Step 2: We frame the inter-cluster data selection as a multi-armed bandit problem, and treat each cluster as one arm. The data influence approximations are considered as the drawing rewards. A modified UCB algorithm is used to draw samples with limited computing budgets. Step 3: From the drawn samples, we select the top portion of data samples with the highest influence as the selected data subset.

loss decrease on $\mathbf{x}_v^j$ after training on $\mathbf{x}_{tr}^i$ for one step (Pruthi et al., 2020). It can be approximated using the first-order Taylor expansion:

$$\begin{aligned}\mathcal{I}^t(\mathbf{x}_{tr}^i, \mathbf{x}_v^j) &= \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^{t+1}) - \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^t) \\ &\approx \langle \nabla \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^t), (\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t) \rangle,\end{aligned} \tag{1}$$

where $\nabla \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^t)$ is the gradient of $\mathbf{x}_v^j$ with respect to $\boldsymbol{\theta}^t$, and $\langle \cdot, \cdot \rangle$ is the inner product.

Since Adam optimizer is the most commonly used optimizer in the SFT of LLMs, the parameter update $\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t$ can be represented with adapted gradients Γ in Adam. Then, the influence can be approximated as:

$$\mathcal{I}^t(\mathbf{x}_{tr}^i, \mathbf{x}_v^j) \approx -\eta^t \langle \nabla \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^t), \Gamma(\mathbf{x}_{tr}^i; \boldsymbol{\theta}^t) \rangle, \tag{2}$$

where $\Gamma(\mathbf{x}_{tr}; \boldsymbol{\theta}^t) = -\eta^t \frac{\mathbf{m}^t}{\sqrt{\mathbf{v}^t + \epsilon}}$, and $\mathbf{m}^t$ and $\mathbf{v}^t$ are the moving averages of historical gradients and their element-wise square, respectively.

3.1.2 Gradient-based data selection

In previous works (Xia et al., 2024; Wang et al., 2025), the data influence approximation is adopted to select the most beneficial training data samples for the SFT on the target task. As studied in previous work (Wang et al., 2025), to mitigate the selection length bias, the gradients in Equation 2 are normalized before calculating the inner product as in:

$$\widetilde{\mathcal{I}}^t(\mathbf{x}_{tr}^i, \mathbf{x}_v^j) \approx \langle \frac{\nabla \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^t)}{\|\nabla \mathcal{L}(\mathbf{x}_v^j; \boldsymbol{\theta}^t)\|}, \frac{\Gamma(\mathbf{x}_{tr}^i; \boldsymbol{\theta}^t)}{\|\Gamma(\mathbf{x}_{tr}^i; \boldsymbol{\theta}^t)\|} \rangle. \tag{3}$$

Wang et al. (2025) point out that the indication of one-step influence approximation has a declining trend during the training process. Thus, they propose to dynamically recompute the data influence approximation and update the selected data subset after every training period. Instead, LESS (Xia et al., 2024) uses a simulation training with a randomly selected data subset to obtain multiple model checkpoints and performs data selection only once with the aggregated one-step data influence approximations over all checkpoints.

In each selection, the data influence approximation $\widetilde{\mathcal{I}}(\mathbf{x}_{tr}^i, \mathbf{x}_v^j)$ is aggregated over all validation data samples. Specifically, $\widetilde{\mathcal{I}}(\mathbf{x}_{tr}^i, \mathbf{x}_v^j)$ is first averaged within each subtask, then the maximum among subtasks is chosen as the data influence approximation $\widetilde{\mathcal{I}}(\mathbf{x}_{tr}^i)$ over the validation dataset. After that, the top $p\%$ of training data samples with the highest data influence approximations are selected to form the selected data subset.

3.2 Reduce computation consumption with clustering

To reduce the computation consumption in gradient-based data selection, one simple way is

to reduce the number of checkpoints used to calculate the one-step data influence approximation, that is, extending the interval between every two calculations. However, the decreasing long-time selection effectiveness phenomenon illustrated in previous work (Wang et al., 2025) shows that too long intervals will lead to highly inaccurate influence approximation, which potentially hinders the data selection. Hence, we dedicate our efforts to reducing the number of gradients needed to be computed at each calculation of one-step data influence approximation.

From Equation 3, the one-step data influence approximation $\widetilde{\mathcal{I}}^t(\mathbf{x}_{tr}^i, \mathbf{x}_v^j)$ is the cosine similarity between the gradients of the training data sample $\mathbf{x}_{tr}^i$ and the validation data sample $\mathbf{x}_v^j$. Then, an intuition is that **the gradients of two training data samples with higher cosine similarity will have similar degrees of cosine similarity with the gradient of the same validation data sample**. Based on this intuition, we perform clustering on the gradients of all training data samples according to their cosine similarities with respect to the pretrained model parameter $\boldsymbol{\theta}^0$. Our experimental observations show that the clusters could remain relatively tight through the model training process, which is discussed in Appendix B.

Then, the data samples with the highest influence should be concentrated in a few clusters. Considering each cluster as a distribution over the data influences, we can allocate our computing resources to the clusters according to their probability of containing training data samples with high influences. In this way, we avoid the calculation over a large number of low-influence training data samples, saving a large portion of computing resources.

3.3 Inter-cluster data selection with UCB algorithm

After obtaining k clusters $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ for training data samples, our next step is to maximize the overall probability of finding the training data samples with the highest influences by allocating the computing budget among different clusters, which we call it inter-cluster data selection problem with constrained computing budget:

$$
\begin{aligned}
B^* = &\arg\max \sum_{c=1}^{k} b_c P_{\widetilde{\mathcal{I}}(\mathbf{x}_{tr})\sim\mathbf{P}_c}(\widetilde{\mathcal{I}}(\mathbf{x}_{tr}) \geq T), \\
s.t. &\sum_{c=1}^{k} b_c = \mathcal{B}, \\
&\forall c \in \{1, ..., k\}, 0 \leq b_c \leq |C_c|,
\end{aligned}
\tag{4}
$$

where $B = \{b_1, b_2, ..., b_k\}$ is an allocation of the computing budget $\mathcal{B}$, $\widetilde{\mathcal{I}}(\mathbf{x}_{tr})$ is the data influence approximation of $\mathbf{x}_{tr}$, $\mathbf{P}_c$ is the distribution of data influences contained in cluster C_c, and T is the lowest influence of the actual top $p\%$ training data samples with the highest influences.

One challenge exists as the distribution $\mathbf{P}_c$ of each cluster is unknown. Hence, the estimation of $\mathbf{P}_c$ needs to be conducted spontaneously with inter-cluster data selection. This problem setting is very similar to the well-known multi-armed bandit problem (Slivkins et al., 2019). Specifically, each cluster can be considered as one arm with an unknown distribution. Once an arm is drawn in each round, a training data sample will be randomly chosen from the corresponding cluster. Then, its data influence approximation will be calculated as the drawing reward in this round. In this way, the number of drawing rounds is the computing budget consumed by the calculation of data influence approximations. Consequently, optimizing the objective in Equation 4 means maximizing the total drawing rewards.

To solve this problem, we adapt the commonly used Upper Confidence Bound (UCB) algorithm (Auer et al., 2002). The core idea of the UCB algorithm is to estimate an upper confidence bound U_c for each arm that corresponds to cluster C_c in our setting. At each drawing round d, the cluster C_d^* with the maximum estimated upper confidence bound U_d^* is chosen to be drawn; then, the reward of drawing C_d^* is acquired and used to update U_d^*. By repeating this process, the estimated upper confidence bound of cluster C_d will be closer to the actual expected reward of C_d. Hence, the cluster C^* with the highest expected reward will be allocated the most computing budget, while the clusters with lower expected rewards will be allocated less computing budget. The modifications we made to the classic UCB algorithm are two-folds: uppper confidence bound estimation with historical reward information and the cold start period.

Upper confidence bound estimation with historical reward information In the classic UCB algorithm (Auer et al., 2002), the upper confidence bound is the upper bound of the confidence interval for the estimation of the mean of each arm. Since our objective in Equation 4 pays more attention to the probability larger than a certain threshold than the mean of the distribution, we record all historical drawing rewards and use them to estimate the distribution of each cluster. Since the actual T is also unknown, directly estimating $P_{\widetilde{\mathcal{I}}(\mathbf{x}_{tr})\sim\mathbf{P}_c}(\widetilde{\mathcal{I}}(\mathbf{x}_{tr}) \geq T)$ could be challenging. Instead, we compare $\hat{T}_c$ for each cluster such that $P^c_{\widetilde{\mathcal{I}}(\mathbf{x}_{tr})\sim\mathbf{P}_c}(\widetilde{\mathcal{I}}(\mathbf{x}_{tr}) \geq \hat{T}_c)$ is approximately equal for all clusters. In practice, we use the estimated mean $\hat{\mu}_c$ and standard deviation $\hat{\sigma}_c$ from the historical drawing rewards to compute $\hat{T}_c$ and consider it as the upper confidence bound U_c of cluster C_c:

$$U_c = \hat{T}_c = \hat{\mu}_c + \beta * \hat{\sigma}_c, \tag{5}$$

where β is a hyperparameter practically set to 1.

Cold start period At the initial stage of the UCB algorithm, the insufficient historical drawing rewards might result in the bad estimation of the upper confidence bounds and large regrets in the objective optimization. Hence, we apply a cold start in our UCB algorithm, which allocates a small portion $p_{cs}\%$ of the computing budget among all clusters proportional to the cluster size. In this starting period, each cluster is drawn multiple times to obtain a set of rewards. Then, this set of rewards can provide a basic estimation of the influence distribution of the corresponding cluster. Since the cold start also accounts for the total computing budget, the cold start ratio affects the trade-off between exploration and exploitation in our UCB algorithm. Specifically, higher cold start ratio means more budgets are allocated to draw from all clusters without specific selection, which improves the exploration of the algorithm; lower cold start ratio means more budgets are allocated to the UCB drawing process, which tends to exploit the clusters with higher estimated upper confidence bounds. Please refer to Appendix C for more discussion. After the cold start period, the algorithm starts to choose the cluster with the largest estimated upper confidence bound at each drawing round.

3.4 Efficient gradient-based data selection framework

Combining clustering and inter-cluster data selection with the UCB algorithm, we propose our efficient gradient-based data selection framework **ClusterUCB**, as shown in Figure 1. Specifically, this framework first clusters all training data samples according to the cosine similarities of their gradients computed with respect to the pretrained model. At each time of data selection, the inter-cluster data selection with the UCB algorithm will be applied under a predefined computing budget.

Since there are still regrets that exist in the drawing process, the computing budget is usually set to be larger than the number of training data samples needed in the end. As the final step, we sort the influence approximations calculated in inter-cluster data selection from high to low and output the top number of corresponding training data samples as our final selected data subset.

4 Experiments

4.1 Experimental setup

Baselines **Random** is to train the model with a randomly selected data subset. **LESS** (Xia et al., 2024) uses simulation training with randomly selected data to acquire multiple model checkpoints, aggregates the one-step data influence approximations of all training data samples with respect to these checkpoints, and chooses the top $p\%$ training data samples with the highest aggregated influence approximations as final selected data subset. **Dynamic** (Wang et al., 2025) directly uses one-step data influence approximations to select top $p\%$ training data samples, but the selection process will be repeated periodically, leading to dynamically updated data subsets through the model training process. To show the effectiveness of our modified UCB algorithm with the same computing budgets, we also implement two vanilla baselines **LESS-Rerank** and **Dynamic-Rerank**, which randomly choose $\mathcal{B}$ training data samples to compute their influence approximations and select the top $p\%$ high-influence data samples among them.

Implementation details Following *LESS* and *Dynamic*, We use LLaMA-2-7B (Touvron et al., 2023) as our pretrained model and set the selection ratio $p\%$ to 5%. The pretrained model is trained for four epochs with an AdamW optimizer. The learning rate is 2e-5 with linear decay. For *Random*, 5%

Methods	Budget	MMLU	TydiQA	GSM8k	HumanEval	Avg.	Δ
Random	-	45.3 (0.5)	48.5 (0.6)	19.7 (0.1)	16.5 (0.8)	32.5	-
LESS	100%	47.0 (0.5)	53.2 (1.1)	27.3 (0.6)	17.7 (0.9)	36.3	-
LESS-Rerank	20%	45.9 (0.2)	52.2 (0.3)	23.8 (0.1)	16.3 (0.3)	34.6	↓ 1.7
LESS-ClusterUCB	20%	**47.8 (0.8)**	**53.7 (1.3)**	**28.0 (1.0)**	**17.6 (0.7)**	**36.8**	↑ **0.5**
Dynamic	100%	47.8 (0.3)	57.6 (0.8)	27.5 (1.2)	19.2 (0.4)	38.0	-
Dynamic-Rerank	20%	46.7 (0.4)	54.9 (0.9)	26.7 (0.6)	**17.9 (0.7)**	36.6	↓ 1.4
Dynamic-ClusterUCB	20%	**47.9 (0.6)**	**57.4 (0.6)**	**27.4 (0.9)**	17.7 (0.4)	**37.6**	↓ **0.4**

Table 1: The results of ClusterUCB and baselines on four commonly-used benchmarks. All experiments are repeated with three random seeds. Δ denotes the difference of average performance between budget-constrained methods and their full-budget counterparts. **Bold** means the best results achieved by budget-constrained methods.

of the training data samples are randomly selected to train the model. For *LESS* and *Dynamic*, the implementation is kept the same as described in their original paper: in *LESS*, 5% randomly selected training data samples are used for simulation training to obtain four checkpoints after each epoch; in *Dynamic*, one-step data selection is performed at the beginning of each epoch, and 20 warmup steps are performed for the first one-step data selection. For ClusterUCB, we also perform 20 warmup steps and use the resulting checkpoint to compute the gradients of all training data samples. Then, K-means (Hartigan and Wong, 1979) is adopted for clustering. We combine ClusterUCB with *LESS* and *Dynamic* to form two variants, **LESS-ClusterUCB** and **Dynamic-ClusterUCB**, respectively. In *LESS-ClusterUCB*, the reward of each draw is the aggregated one-step data influence approximations as in *LESS*. In *Dynamic-ClusterUCB*, since the gradients used for clustering and the first one-step data selection are the same and complete, we keep the first training epoch the same as that in *Dynamic*, then apply our proposed inter-cluster data selection in the following three one-step data selections. In our main experiments, for *LESS-ClusterUCB*, *Dynamic-ClusterUCB*, *LESS-Rerank*, and *Dynamic-Rerank*, the number of clusters k is 150, the cold start ratio $p_{cs}\%$ is 5%, and the computing budget $\mathcal{B}$ is 20% of the total number of training data samples. All gradients are computed using LoRA (Hu et al., 2022) and projected to 8192-dimensional vectors using Random Projection (Park et al., 2023).

Datasets The training data pool is the mix of eight commonly-used datasets: **Flan-v2** (Longpre et al., 2023) is a large SFT dataset converted from various NLP datasets; **CoT** (Longpre et al., 2023) is a subset of Flan-v2 with chain-of-though; **Dolly** (Conover et al., 2023) is a high-quality instruction-following dataset generated by humans; **Open Assistant v1** (Köpf et al., 2023) is a multi-round chatting datasets generated by human and open-sourced LLMs; **GPT4-Alpaca** (Peng et al., 2023) contains instructions in Alpaca dataset and answers regenerated by GPT-4; **ShareGPT** (Chiang et al., 2023) is a conversation datasets with mixed-quality; **GSM8k train** (Cobbe et al., 2021) is a primary school-level math word dataset; **Code-Alpaca** (Chaudhary, 2023) is a dataset designed for the development of model's coding ability.

Evaluation benchmarks and validation data samples We adopt four commonly used benchmarks covering the general, multilingual, mathematical, and coding abilities of LLMs. **MMLU** (Hendrycks et al., 2021) is a knowledge-based multi-choice QA benchmark including 57 subjects; **TydiQA** (Clark et al., 2020) is a multi-language QA benchmark including nine languages; **GSM8k** (Cobbe et al., 2021) is a math reasoning benchmark evaluating models' mathematical reasoning ability; **HumanEval** (Chen et al., 2021) is a Python coding benchmark evaluating models' code generation ability. The selection and aggregation of validation data samples follows *Dynamic* for all methods: the few-shot samples of MMLU and TydiQA are directly used as validation data samples; 50 and 10 test data samples are randomly selected from GSM8k and HumanEval as validation data samples.

4.2 Main results

The performances of ClusterUCB and baselines on four benchmarks are shown in Table 1. All gradient-based methods outperform *Random* to a large margin on the average performance of four benchmarks, showing that gradient-based methods are effective in selecting suitable data subsets for the targeted fine-tuning of LLMs.

Bgt	MMLU	TydiQA	GSM8k	HE
10%	46.3 (1.1)	55.4 (0.5)	28.8 (0.5)	18.8 (0.6)
20%	47.9 (0.6)	57.4 (0.6)	27.4 (0.9)	17.7 (0.4)
30%	47.3 (0.2)	57.5 (0.5)	27.2 (1.4)	18.3(0.3)

Table 2: The results of *Dynamic-ClusterUCB* with different computing budgets. Bgt and HE are the abbreviations for Budget and HumanEval, respectively.

With the computing budget set to 20%, both *LESS-ClusterUCB* and *Dynamic-ClusterUCB* match their full-budget counterparts *LESS* and *Dynamic* on most benchmarks. These results show that ClusterUCB can reduce the computational consumption of gradient-based data selection methods while maintaining their performance. Although the performance of *Dynamic-ClusterUCB* on HumanEval drops compared to *Dynamic*, in Section 4.3, we find that it could achieve better results when the computing budget is reduced to 10%.

Using the same computing budget, both *LESS-ClusterUCB* and *Dynamic-ClusterUCB* outperform *LESS-Rerank* and *Dynamic-Rerank* on almost all benchmarks, further indicating the effectiveness of the selection strategy used in ClusterUCB.

4.3 Influence of computing budgets

To illustrate the influence of computing budgets, we conduct experiments on *Dynamic-ClusterUCB* with computing budgets $\mathcal{B}$=10%, 20%, and 30%. The cold start ratio $p_{cs}\%$ is 5%, and the number of clusters is 150, as in the main experiments. The results in Table 2 show that different benchmarks have different change patterns along with the computing budget. On MMLU and TydiQA, the performance of *Dynamic-ClusterUCB* is worse when $\mathcal{B}$ is only 10%. $\mathcal{B}$ = 20% is enough since increasing $\mathcal{B}$ from 20% to 30% leads to trivial performance improvements. On the contrary, on GSM8k and HumanEval, the smaller computing budgets tend to result in higher accuracies. The reason might be that the training data samples useful for the improvement of mathematical and coding abilities are spread across only a few clusters. Thus, a small computing budget is sufficient to find the data samples with high influence once our UCB algorithm finds the correct arms. The degradation of performance with the increase of computing budgets on GSM8k and HumanEval might lie in the randomness of data selection and training.

Methods	TydiQA	HumanEval
Random	64.5 (0.2)	38.4 (1.2)
LESS	66.8 (0.5)	36.9 (1.5)
LESS-ClusterUCB	66.7 (0.5)	38.1 (0.9)
Dynamic	67.3 (0.6)	40.0 (1.5)
Dynamic-ClusterUCB	67.8 (0.1)	40.9 (1.2)

Table 3: The results of ClusterUCB and baselines using Qwen2.5-3B as the pretrained model.

4.4 Results on a different model

To further evaluate ClusterUCB on a different model architecture and scale, we conduct experiments using Qwen2.5-3B (Team, 2024) as the pretrained model on TydiQA and HumanEval benchmarks. The implementation details are kept the same as in our main experiments, except that all models are trained for three epochs.

The results are shown in Table 3. Consistent with the results using LLaMA-2-7B as the pretrained model, both *LESS-ClusterUCB* and *Dynamic-ClusterUCB* match their full-budget counterparts *LESS* and *Dynamic* on these two benchmarks, showing the effectiveness of ClusterUCB on different model architectures and scales.

4.5 Hyperparameter analysis

We analyze the impacts of two key hyperparameters in ClusterUCB: the number of clusters k and the cold start ratio $p_{cs}\%$. We adopt two metrics to evaluate the goodness of the selected hyperparameters. One is the sample-level recall rate R_s, and the other one is the influence-level recall rate R_{inf}, as shown in Equation 6 and 7, respectively, where D is the final data subset selected by ClusterUCB and D_{gt} is the actual top portion of training data samples with the highest influence.

$$R_s = \frac{|D \bigcap D_{gt}|}{|D_{gt}|} \tag{6}$$

$$R_{inf} = \frac{\sum_{\mathbf{x}^i_{tr} \in D} \widetilde{\mathcal{I}}(\mathbf{x}^i_{tr})}{\sum_{\mathbf{x}^q_{tr} \in D_{gt}} \widetilde{\mathcal{I}}(\mathbf{x}^q_{tr})}. \tag{7}$$

We compute R_s and R_{inf} on the model checkpoint obtained after warmup training, and the computing budget $\mathcal{B}$ is fixed to be 20% in this section.

4.5.1 Cold start ratio

Set k = 150, we evaluate $p_{cs}\%$ = {0%, 5%, 25%, 50%, 75%, 100%}, as shown in Figure 2.

	Random-Draw		UCB1		UBC-TN		UCB-TH		UCB-Beta	
Tasks	R_s	R_{inf}	R_s	R_{inf}	R_s	R_{inf}	R_s	R_{inf}	R_s	R_{inf}
MMLU	20.14	72.72	26.24	76.76	73.96	96.40	77.00	96.96	**77.24**	**96.97**
TydiQA	19.38	74.74	23.12	77.21	59.08	92.96	68.96	**95.56**	**69.03**	95.52
GSM8k	25.84	45.87	59.31	90.05	90.64	99.13	93.12	99.47	**93.75**	**99.52**
HumanEval	22.63	58.68	24.72	61.01	55.96	86.95	**71.61**	**93.83**	71.50	93.75

Table 4: The sample-level and influence-level recall rates of different upper confidence bound evaluation metrics. **Bold** indicates the best results for each task.

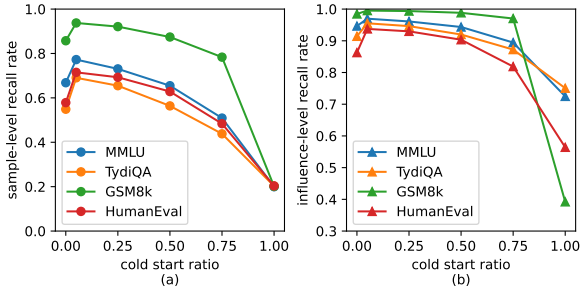


Figure 2: The sample-level and influence-level recall rates with different cold start ratios.

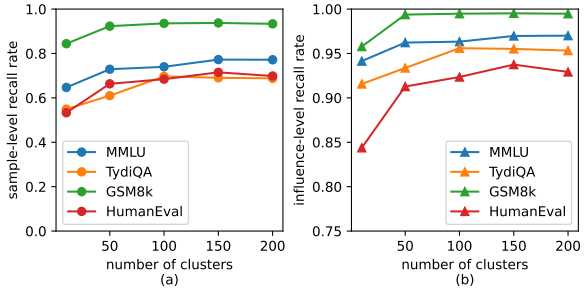


Figure 3: The sample-level and influence-level recall rates with different numbers of clusters.

Sufficient historical reward information is necessary. When $p_{cs}\% = 0\%$, both R_s and R_{inf} are low for all tasks. But with $p_{cs}\%$ increases to 5%, R_s and R_{inf} improve obviously. It shows that sufficient historical reward information with a cold start is necessary in the initial stage of inter-cluster data selection.

Our UCB algorithm is effective in inter-cluster data selection. When $p_{cs}\%$ increases to 100%, the inter-cluster data selection degrades to simply allocating computing budgets proportional to the cluster size, and R_s and R_{inf} become extremely low. It illustrates that our UCB algorithm is effective in inter-cluster data selection.

The trade-off between exploration and exploitation. With the increase of $p_{cs}\%$ from 5%, R_s and R_{inf} gradually decrease, showing that only a small portion of the computing budget should be used for the cold start. The ratio of cold start also controls the trade-off between exploration and exploitation in the UCB algorithm. These results indicate that exploration is necessary and important in our UCB algorithm, but too much exploration could hinder the algorithm's performance. We refer to Appendix C for further discussion about the distribution of data selection among clusters.

4.5.2 Number of clusters

Set $p_{cs}\% = 5\%$, we evaluate $k = \{10, 50, 100, 150, 200\}$. The results are shown in Figure 3. When the number of clusters is as small as 10, both R_s and R_{inf} are the worst for all tasks, indicating that too small number of clusters might not be able to fully separate training data samples into groups with similar gradients. Increasing the number of clusters from 10 to 50, R_s and R_{inf} show obvious improvement. Further increasing the number of clusters, the improvements become less observable, and R_s and R_{inf} tend to be stable. It also indicates that ClusterUCB is not sensitive to the number of clusters, as long as it is not too small.

4.6 Comparison of different upper confidence bound evaluation metrics

In Section 3.3, we evaluate the upper confidence bound U_c as the estimated influence threshold $\hat{T}_c$ that corresponds to the same probability. An alternative is to directly estimate $P_{\widetilde{\mathcal{I}}(\mathbf{x}_{tr})\sim\mathbf{P}_c}(\widetilde{\mathcal{I}}(\mathbf{x}_{tr}) \geq T)$. One estimation is the ratio of drawing with rewards larger than T, which we call **UCB-TH**. We could also consider each cluster distribution $\mathbf{P}_c$ as a Gaussian distribution with the mean and standard deviation estimated from the historical reward values $\widetilde{\mathbf{P}}_c \sim \mathcal{N}(\hat{\mu}_c, \hat{\sigma}_c)$, and compute $P_{\widetilde{\mathcal{I}}(\mathbf{x}_{tr})\sim\widetilde{\mathbf{P}}_c}(\widetilde{\mathcal{I}}(\mathbf{x}_{tr}) \geq T)$. We call this estimation **UCB-TN**. Since T is unknown, we estimate it as the lowest influence in the top $p/\mathcal{B}$ portion of all

historical reward values of all clusters in the current round.

Keeping all hyperparameters and computing budget the same as in the main experiments, we compare *UCB-TH* and *UCB-TN* with the evaluation metrics used in our main experiments (**UCB-Beta**). We also compare them with two baselines: **Random-Draw** that randomly chooses an arm to draw at each round, and a classic UCB algorithm **UCB1** (Auer et al., 2002).

The results in Table 4 show that *UCB-Beta* and *UCB-TH* achieve the best results among all tasks, and the former is slightly better than the latter in most tasks. It indicates that *UCB-Beta* and *UCB-TH* might be equivalent in our setting. *UCB-TN* is worse than *UCB-Beta* and *UCB-TH*, indicating that using a Gaussian distribution to fit the cluster distribution might be inaccurate. Although *UCB1* performs better than *Random-Draw*, it is far worse than *UCB-Beta*, *UCB-TH*, and *UCB-TN*, showing that only estimating the mean of the distribution of each cluster could not solve the inter-cluster data selection problem.

5 Conclusion

In this paper, we aim to reduce the computational consumption used in gradient-based SFT data selection for LLMs. Our proposed framework first performs clustering over the training data pool based on the intuition that training data samples with similar gradients would have similar influences on target loss optimization. Then, we frame the inter-cluster data selection as a computing budget allocation problem which is similar to the multi-armed bandit problem, and modify the UCB algorithm to solve it. Combined with the state-of-the-art gradient-based data selection methods, experimental results show that our proposed framework can match the original methods while greatly reducing the computing consumption.

Limitations

While our proposed framework has been proven to be efficient in saving computing resources, it is also essential to consider its limitations that may be improved in the future. As stated by Wang et al. (2025), the gradient-based data selection methods only consider the influence of single data samples, neglecting the mutual influences within the selected data subsets. With the clusters generated, the inter-cluster data selection could consider groups of data samples as the arm-drawing rewards, which is the next step of our work. Data diversity of the selected subset might also be improved by balancing the number of samples selected in each cluster (Zhang et al., 2025). Moreover, we use K-means, the simplest clustering algorithm, in this paper. Better clustering might also improve the performance of our proposed framework.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China No.62372459 and No.62302503, NUDT Youth Independent Innovation Science Fund Project Grant No.ZK23-15, the Open Research Fund from State Key Laboratory of High Performance Computing of China Grant No.202401-09, and 10th Youth Talent Support Program of the China Association for Science and Technology.

References

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. 2024. A survey on data selection for language models. *Trans. Mach. Learn. Res.*, 2024.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290v3*.

Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. 2021. Kernel operations on the gpu, with autodiff, without memory overflows. *J. Mach. Learn. Res.*, 22:74:1–74:6.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. 2019. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 32.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela

Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374v2*.

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. 2024. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36.

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

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 ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168v2*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, 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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3029–3051. Association for Computational Linguistics.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 177–198. Association for Computational Linguistics.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. 2024. DsDm: Model-aware dataset selection with datamodels. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12491–12526. PMLR.

Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. 2023. Simfluence: Modeling the influence of individual training examples by simulating training runs. *arXiv preprint arXiv:2303.08114v1*.

Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.

John A Hartigan and Manchek A Wong. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 10864–10882. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622v1*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations–democratizing large language model alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Po-Nien Kung, Fan Yin, Di Wu, Kai wei Chang, and Nanyun Peng. 2023. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1813–1829. Association for Computational Linguistics.

Huawei Lin, Jikai Long, Zhaozhuo Xu, and Weijie Zhao. 2024. Token-wise influential training data retrieval for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 841–860. Association for Computational Linguistics.

Qingyi Liu, Yekun Chai, Shuohuan Wang, Yu Sun, Keze Wang, and Hua Wu. 2024. On training data influence of gpt models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 3126–3150. Association for Computational Linguistics.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Fei Mi, Baojun Wang, Weichao Wang, Lifeng Shang, and Qun Liu. 2023a. Self: Language-driven self-evolution for large language model. *arXiv preprint arXiv:2310.00533v4*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023b. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models.

Yanzhou Pan, Huawei Lin, Yide Ran, Jiamin Chen, Xiaodong Yu, Weijie Zhao, Denghui Zhang, and Zhaozhuo Xu. 2025. Alinfik: Learning to approximate linearized future influence kernel for scalable third-parity LLM data valuation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 11756–11771. Association for Computational Linguistics.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning (ICML)*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277v1*.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.

Aleksandrs Slivkins and 1 others. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288v2*.

Fanqi Wan, Xinting Huang, Tao Yang, Xiaojun Quan, Wei Bi, and Shuming Shi. 2023. Explore-instruct: Enhancing domain-specific instruction coverage through active exploration. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9435–9454. Association for Computational Linguistics.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and 1 others. 2023a. How far can camels go? exploring the state of instruction tuning on open resources. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zige Wang, Wanjun Zhong, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023b. Data management for training large language models: A survey. *arXiv preprint arXiv:2312.01700v3*.

Zige Wang, Qi Zhu, Fei Mi, Yasheng Wang, Haotian Wang, and Lifeng Shang. 2025. Dynamic data selection with normalized gradient-based influence approximation for targeted fine-tuning of llms. *Knowledge-Based Systems*, 327:114144.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. 2025. Harnessing diversity for important data selection in pretraining large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L Zhang. 2024. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25*

May, 2024, Torino, Italy, pages 16776–16789. ELRA and ICCL.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Implementation details

A.1 Training datasets

We use the same training datasets as (Wang et al., 2025) did. The number of instances in each dataset are shown in Table 5. The number of averaged completion tokens in our training data pool is 189.1. Setting the selection ratio to 5%, the selected data subset contains 20,387 instances.

A.2 Training details

Following Xia et al. (2024) and (Wang et al., 2025), we adopt the parameter-efficient fine-tuning method LoRA (Hu et al., 2022) in all our experiments. The rank of LoRA module is 128, the value of α is 512, and the learnable LoRA matrics are applied to all attention matrices. Under this configuration, there are 134,217,728 trainable parameters in LLaMA-2-7B accounting for 1.95% of the original parameters, and 58,982,400 trainable parameters in Qwen2.5-3B accounting for 1.88% of the original parameters. All of our experiments are conducted using 8 Tesla V100 GPUs.

A.3 Evaluation details

We use the evaluation code toolkit provided by Open-Instruct (Wang et al., 2023a). On MMLU, the evaluation metric is the exact match of the first token in models' completion and the ground truth answer, we perform evaluation in a 5-shot setting and average over the 57 subtasks; On TydiQA, gold passage and 1-shot are adopted, and the performance is evaluated as the F1 score of the models' completions and the ground truth answers and averaged over nine languages; On GSM8k, 8-shot is adopted, and the final number in models' completion is extracted as the final answer to exactly match with the ground truth answer; On HumanEval, we use pass@1 as the evaluation metric and sample 20 completions for each instruction with temperature 0.1.

Dataset	# Instance
Flan v2	99,245
CoT	95,557
Dolly	14,865
Open Asisstant v1	54,626
GPT4-Alpaca	52,002
ShareGPT	63,951
GSM8k train	7,473
Code-Alpaca	20,021
Total	40,7740

Table 5: The number of instances in each dataset used in our experiments.

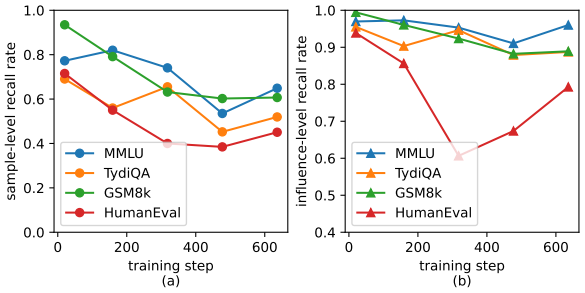


Figure 4: The change of sample-level and influence-level recall rates during the training process.

A.4 Clustering details

We adopt LoRA in our experiments, and all gradients are projected to 8192-dimensional vectors before clustering. In the implementation of clustering, we use an efficient library, PyKeops (Charlier et al., 2021), to perform kernel matrix operations. The training data pool used in our experiments contains 407,740 data samples, so we have 407,740 8192-dimensional vectors as the inputs of clustering. With the number of clusters $k = 150$ and iterations in K-means $N = 20$, the computation time of clustering is 1.978h.

B Declination of the effectiveness of clusters during training

In our proposed framework, we only perform clustering according to the cosine similarities of the gradients of training data samples at the beginning of training. Since the gradients of training data samples would change with the update of model weights, whether the clustering is still effective during the training process would be an essential problem. Thus, we conduct experiments to study the changing trend of the effectiveness of clusters during training.

We again use the sample-level recall rate R_s in

Equation 6 and influence-level recall rate R_{inf} in Equation 7 to evaluate the effectiveness of clusters. We compute R_s and R_{inf} with respect to the checkpoints saved during training with 5% randomly selected data samples. The number of clusters k = 150, the cold start ratio $p_{cs}\%$ = 5%, as in our main experiments. The results are illustrated in Figure 4.

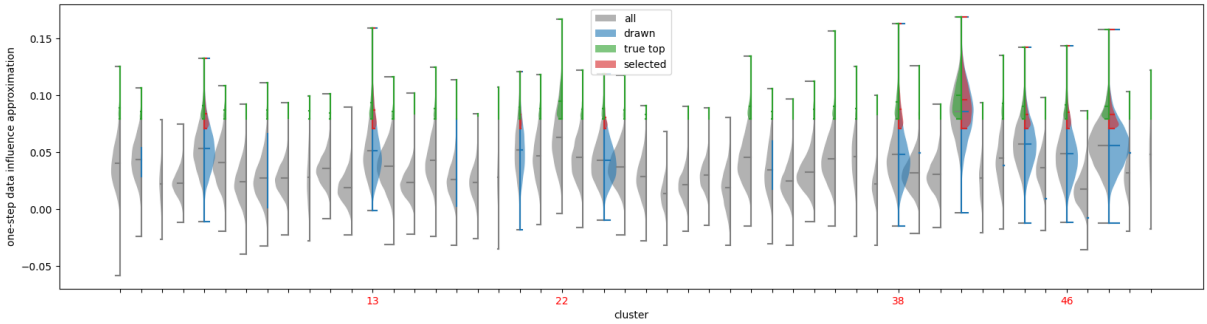
On most benchmarks, R_s and R_{inf} show declining trends when the training step increases. This indicates that updating clustering after certain training steps could lead to better results. However, the updating of clusters also introduces extra computational consumption. Moreover, R_{inf} still remains high in the later stage of training, indicating that using the clusters computed at the initial stage can still select the data samples with relatively high influences. Hence, we choose not to update the clustering in our experiments. Still, we obtain comparable results with methods using the full budget according to Table 1.

C Exploration vs. exploitation of inter-cluster data selection

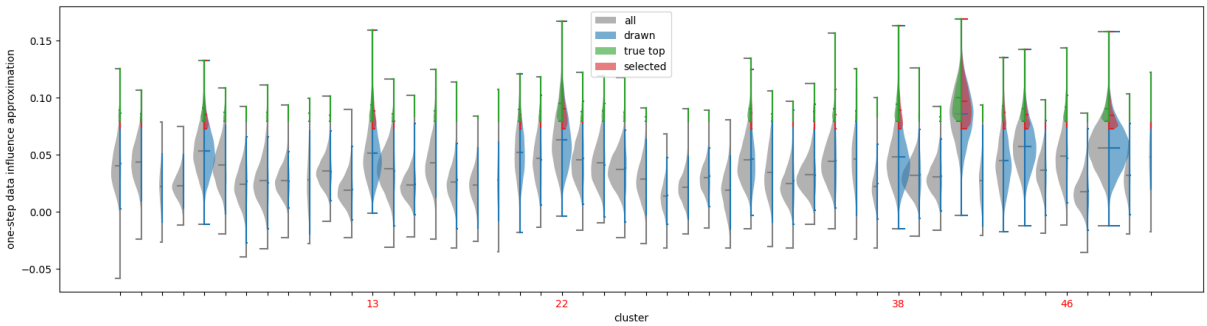
As discussed in Section 4.5.1, the cold start ratio $p_{cs\%}$ controls the trade-off between exploration and exploitation in our UCB algorithm. To further observe the effect of this trade-off, we plot the distribution of the total data samples contained, the data samples drawn in our UCB algorithm, the true top portion of data samples with the highest data influence approximations, and the selected data samples using our modified UCB algorithm within each cluster. For simplicity, we plot with the number of clusters k = 50 on the MMLU benchmark. To compare the effect of different degrees of exploration and exploitation, we plot with the cold start ratio $p_{cs}\%$ = 0%, 5%, and 50%, as shown in Figure 5a, 5b and 5c, respectively.

When $p_{cs}\%$ = 0%, the modified UCB algorithm does not adopt the cold start strategy and tends to assign most of the computing budget to exploitation. Accordingly, the distribution of the drawn data samples in Figure 5a is more concentrated on a few clusters, which does not cover many clusters with high-influence data samples. Assigning a small budget to random exploration with $p_{cs}\%$ = 5%, the evaluation of each cluster is more accurate in our UCB algorithm, leading to larger probability to find clusters with more high-influence data samples, e.g., cluster No. 22 and 46 in Figure 5b. Continue increasing $p_{cs}\%$ to 50%, more budget is spend on exploration, resulting in insufficient budget for exploitation, the modified UCB algorithm is more likely to miss high-influence data samples even though it can hit the corresponding clusters, e.g., cluster No. 13 and 38 in Figure 5c.

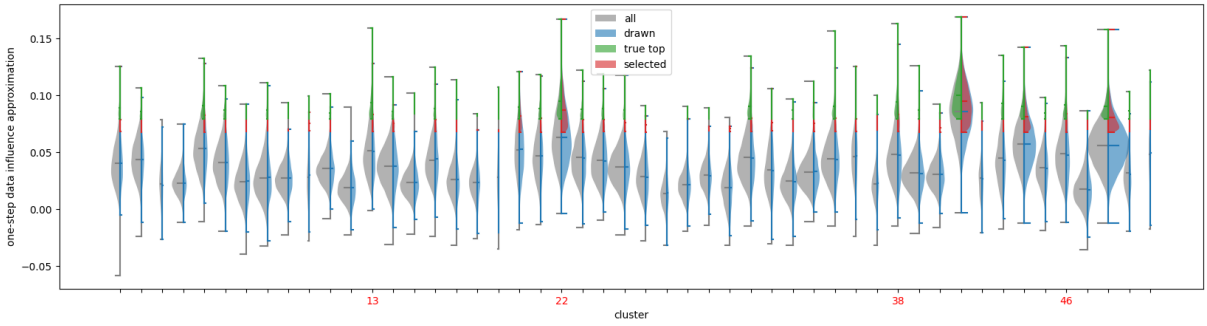
Thus, the distribution of inter-cluster data selection is consistent with our experimental results in Section 4.5.1, that a cold start with random exploration is necessary in the inter-cluster data selection, but spending too much budget on exploration could be harmful and lead to worse performance.



(a) cold start ratio = 0%



(b) cold start ratio = 5%



(c) cold start ratio = 50%

Figure 5: Data distributions among clusters with different cold start ratios on MMLU benchmark. Each violin graph represents one cluster. For each cluster, the gray half is the total data samples contained in this cluster; the blue half is the data samples drawn from this cluster in our UCB algorithm; the green half is the true top portion of data samples with the highest influences contained in this cluster; the red half is the selected data samples from this cluster using ClusterUCB. The width of each half represents the number of data samples in this half.