Efficient Pretraining Data Selection for Language Models via Multi-Actor Collaboration

Tianyi Bai[1,2], Ling Yang[3], Zhen Hao Wong[3], Fupeng Sun[5],
Xinlin Zhuang[2], Jiahui Peng[2], Chi Zhang[2], Lijun Wu [2],
Jiantao Qiu[2*†], Wentao Zhang[2,3,4*], Binhang Yuan[1*], Conghui He[2*],
[1]Hong Kong University of Science and Technology, [2]Shanghai Artificial Intelligence Laborator,
[3]Peking University, [4]Zhongguancun Academy, [5]Imperial College London

Correspondence: wentao.zhang@pku.edu.cn, biyuan@ust.hk, {heconghui, qiujiantao}@pjlab.org.cn

Abstract

Efficient data selection is crucial to accelerate the pretraining of language model (LMs). While various methods have been proposed to enhance data efficiency, limited research has addressed the *inherent conflicts* between these approaches to achieve optimal data selection for LM pretraining. To tackle this problem, we propose a *multi-actor collaborative data selection* mechanism: each data selection method independently prioritizes data based on its criterion and updates its prioritization rules using the current state of the model, functioning as an independent actor for data selection; and a console is designed to adjust the impacts of different actors at various stages and dynamically integrate information from all actors throughout the LM pretraining process. We conduct extensive empirical studies to evaluate our multi-actor framework. The experimental results demonstrate that our approach significantly improves data efficiency, accelerates convergence in LM pretraining, and achieves an average relative performance gain up to 10.5% across multiple language model benchmarks compared to the state-of-the-art methods. Code and checkpoints are publicly released at https://github.com/Relaxed-System-Lab/multi-actor-data-selection.

1 Introduction

Efficient data selection is crucial for the pretraining of language model (LMs), as the quality of training data significantly impacts the statistical efficiency of the training procedure and the model performance (Brown et al., 2020; Du et al., 2022; Chowdhery et al., 2023). Recently, we have witnessed numerous approaches, such as filtering high-quality data (Xie et al., 2023b; Wettig et al., 2024), mixing data from multiple domains (Xie et al., 2023a; Liu et al., 2024), and selecting data that optimally boosts downstream task performance dynamically (Engstrom et al., 2024; Yu et al., 2024), which aim to improve data efficiency by prioritizing more informative training samples. However, these methods often operate independently or in isolated settings, limiting their potential when integrated into a collaborative framework. In this work, we want to explore *how to effectively, flexibly, and robustly combine these advanced data selection techniques through the dynamic pretraining process*, addressing the challenges of optimizing data efficiency for LM pretraining at scale.

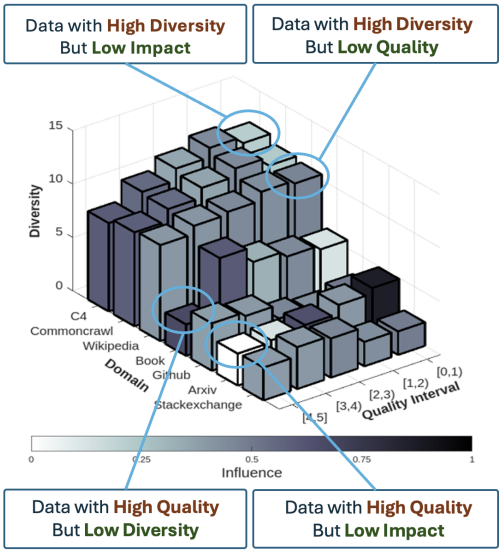


Figure 1: This figure shows the 627B token distribution by quality, domain, topic diversity, and model impact at step 1500. Each bar represents a subset, highlighting trade-offs between diversity, quality, and influence.

Nowadays, various heuristic methods have been proposed to provide measurements for the data samples used during LM pre-training, aiming to optimize data efficiency by selecting or weighting the most informative training examples. However, we observe that integrating multiple data selection and mixing strategies presents significant challenges due to their *inherent conflicts*. For example, high-

*Corresponding authors.
†Project lead.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9465–9491
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

quality data identified by scoring functions may not align with data that strongly impact model performance as measured by influence functions (Engstrom et al., 2024); similar conflicts also exists between other methods — further details are enumerated in §2. These observations actually motivate us to launch a systematic discussion about how to effectively integrate these methods during the dynamic pretraining process that provides superior data efficiency for LM pretraining.

Effectively integrating multiple data selection methods into a unified framework presents significant challenges. While each method may individually offer benefits, combining them requires navigating an exponential search space of possible configurations, which quickly becomes infeasible at scale. This challenge is further exacerbated in online data selection settings, where decisions must be made dynamically during training. Unlike offline approaches that rely on static, precomputed heuristics—such as classifier-based scoring, domain weighting, or topic sampling (Brown et al., 2020; Team, 2024; Xie et al., 2023a)—online methods aim to adaptively select data based on the model's evolving learning state. For example, MATES (Yu et al., 2024) continuously probes the pretraining model to estimate the influence of individual data points and trains a lightweight data influence model to predict which samples will be most beneficial at each stage of training. Similarly, DSDM (Engstrom et al., 2024) formulates data selection as an optimization problem, selecting data subsets that maximize performance on downstream tasks without relying on predefined quality metrics. Despite their effectiveness, these methods incur substantial computational overhead, as they require labeling or evaluating the entire dataset at each stage of training—an approach that is impractical for large-scale pretraining. Consequently, the core challenge is to develop an online integration framework that preserves the adaptivity and performance benefits of these techniques while remaining computationally efficient and scalable.

To address these challenges, we conduct a case study to identify the inherent conflicts for existing data selection methods and provide a multi-actor collaborative data selection framework to resolve this issue. Our multi-actor framework is inspired by the classical definition of intelligence outlined by (Russell and Norvig, 2016), where an actor is defined as an entity that perceives a state and maps the observed state to actions. In our approach, data selection is achieved by the collaboration of these actors. Our contributions can be summarized as:

Contribution 1: Our case study on the SlimPajama reveals conflicts among four data selection metrics in LM pretraining. Despite these conflicts, prior studies (Wettig et al., 2024; Xie et al., 2023a; Yu et al., 2024) show that even a single metric can effectively guide training, highlighting the need for better integration of these approaches.

Contribution 2: We propose a novel multi-actor collaborative data selection framework (§3), where each method acts as an independent scorer for training data. An actor console integrates these scores to optimize selection, and a dynamic collaboration mechanism adjusts actor contributions throughout training, enhancing flexibility and data efficiency.

Contribution 3: Extensive experiments demonstrate that: (1) our method significantly improves data efficiency, accelerating LM convergence and achieving up to a 10.5% performance boost over baselines (§4.1); and (2) ablation studies confirm that key components of our framework are essential for these gains (§4.2).

2 Case Study - Inherent Conflicts in Data Selection

In this section, we present several observations derived from the SlimPajama datasets (Soboleva et al., 2023), which reveal some inherent conflicts for different data selection measurements. To conduct this case study, we first label all data from the SlimPajama datasets using the quality scorer FineWeb-Edu (Lozhkov et al., 2024). We then divide the data into subsets based on domain and quality ranges. From each subset, we uniformly sample data to assess topic diversity, i.e., the topic classification of the sampled data according to our methods. We analyze this diversity by examining the topic distribution within each subset. Additionally, we compute the normalized influence of the data on a pretrained 1.3B model at the 1500th step using influence functions to evaluate the data's impact on the model (Engstrom et al., 2024). Figure 1 illustrates the results, which presents a bar chart representing four dimensions: quality, domain, topic diversity, and influence on the pretrained model. The **x-axis** shows data quality, with higher intervals reflecting better scores from the FineWeb-Edu quality scorer. The **y-axis** indicates the dataset's domain, while the **z-axis** shows topic diversity within each subset, with taller bars indicating more diversity. The **color gradient** represents influence on

the model, with darker shades showing greater impact. From this analysis, we highlight the following interesting observations:

- *High-quality data identified by the quality scorer may not have a significant impact on model performance.* For example, ArXiv documents rated between 4 and 5 by the scorer are considered high-quality. However, at the 1500th training step, they exert minimal impact on the model according to the influence functions, revealing a discrepancy between data quality and model impact. This observation is consistent with the previous discussion in (Engstrom et al., 2024).
- *High-quality data may exhibit low topic diversity.* Documents in the Book domain with a quality score of 4 to 5 are classified as high-quality by the scorer. Nevertheless, 85% of these documents belong to the same topic, indicating a lack of diversity.
- *Data with high topic diversity may not strongly influence model performance.* Documents from the C4 domain display considerable topic diversity. However, at the 1500th training step, they have limited impact on the model as measured by the influence functions, suggesting a conflict between diversity and model influence.
- *Data with high topic diversity can be low quality.* Wikipedia documents show substantial topic diversity, which benefits the topic classifier. However, some of these documents are rated as low-quality by the quality classifier, revealing a trade-off between diversity and quality.

We believe this inherent conflict illustrates that a naive ensemble of these mechanisms may lead to poor performance in terms of data efficiency for LM pretraining, which motivates the design and implementation of our multi-actor collaborative framework in §3.

3 Collaborative Data Selection

In this section, we present the formalization of the data selection problem in §3.1, outline the overall framework of our methods in §3.2, and detail the actor initialization and update in §3.3, along with the collaborative mechanism in §3.4.

3.1 Problem Formulation

We follow the definition of the data selection problem in (Engstrom et al., 2024) and (Yu et al., 2024) with slight modification. The objective for data selection is to choose a subset of size k from the entire pretraining dataset in such a way that the trained model's loss on downstream tasks is minimized. Let $\mathcal{O}$ represent an optimization algorithm that maps a training dataset to a trained model. The optimal subset $\mathcal{D}_k^*$ of the pretraining dataset $\mathcal{D}$ can be expressed as:

$$\mathcal{D}_k^* \coloneqq \underset{\mathcal{D}_k \subset \mathcal{D}, |\mathcal{D}_k|=k}{\arg\min} \mathcal{L}(\mathcal{D}_k \mid \mathcal{M}, \mathcal{T}_{\text{eval}}), \tag{1}$$

where

$$\mathcal{L}(\mathcal{D}_k \mid \mathcal{M}, \mathcal{T}_{\text{eval}}) \coloneqq \mathbb{E}_{x \sim \mathcal{T}_{\text{eval}}} [\ell(x; \mathcal{O}(\mathcal{M}, \mathcal{D}_k))]$$

denotes the expected loss (e.g., cross-entropy loss) for model $\mathcal{M}$ on downstream task $\mathcal{T}_{\text{eval}}$. Minimizing this objective directly is computationally challenging. Given that the real downstream tasks are unknown during model training, prior works have approximately optimized this problem by minimizing the loss on selected reference tasks D_{ref} (e.g. LAMBADA (Paperno et al., 2016), SQuAD (Rajpurkar, 2016) and Jeopardy (Tunguz, 2019) in (Engstrom et al., 2024)). Specifically, they train proxy models to compute one-step training loss (Yu et al., 2024) or influence functions (Engstrom et al., 2024) on the reference tasks to approximate the true loss. However, this approach heavily depends on the selection of the reference tasks, while the chosen reference tasks may not be fully representative of all potential downstream tasks.

To avoid this obstacle, we do not directly minimize the loss on the reference tasks. Instead, we view this loss as a *reward signal* that guides the update of predefined data selection methods. Concretely, we define a reward function $R(\mathcal{D}_k \mid \mathcal{M}, \mathcal{T}_{\text{ref}})$, where the reward is based on the performance gain of current model $\mathcal{M}$ trained on the subset $\mathcal{D}_k$ and evaluated on the reference tasks $\mathcal{T}_{\text{ref}}$. Then our optimization goal becomes maximizing this reward over time, as:

$$\mathcal{D}_k^* = \underset{\mathcal{D}_k \subset \mathcal{D}, |\mathcal{D}_k|=k}{\arg\max} \mathbb{E}\left[R(\mathcal{D}_k \mid \mathcal{M}, \mathcal{T}_{\text{ref}})\right] \tag{2}$$

where

$$R(\mathcal{D}_k | \mathcal{M}, \mathcal{T}_{\text{ref}}) \coloneqq \mathbb{E}_{x \sim \mathcal{T}_{\text{ref}}} \left[-\ell(x; \mathcal{O}(\mathcal{M}, \mathcal{D}_k))\right].$$

We will rewrite $R(\mathcal{D}_k \mid \mathcal{M}, \mathcal{T}_{\text{ref}})$ as $R(\mathcal{D}_k)$ if there is no ambiguity. In practice, $R(\mathcal{D}_k)$ can be approximated by the weighted average of influence functions (Engstrom et al., 2024; Yu et al., 2024), which is defined by

$$r(x_i) = -\nabla_{\mathcal{M}} \mathcal{L}(\mathcal{T}_{\text{ref}} \mid \mathcal{M})^\top H_{\mathcal{M}}^{-1} \nabla_{\mathcal{M}} \mathcal{L}(x_i \mid \mathcal{M}), \tag{3}$$

where $H_{\mathcal{M}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\mathcal{M}}^2 \mathcal{L}_{\mathcal{M}}(x_i \mid \mathcal{M})$ is the Hessian and its positive definite. Details of calculating influence functions for pretraining data point can be found in §A.9.

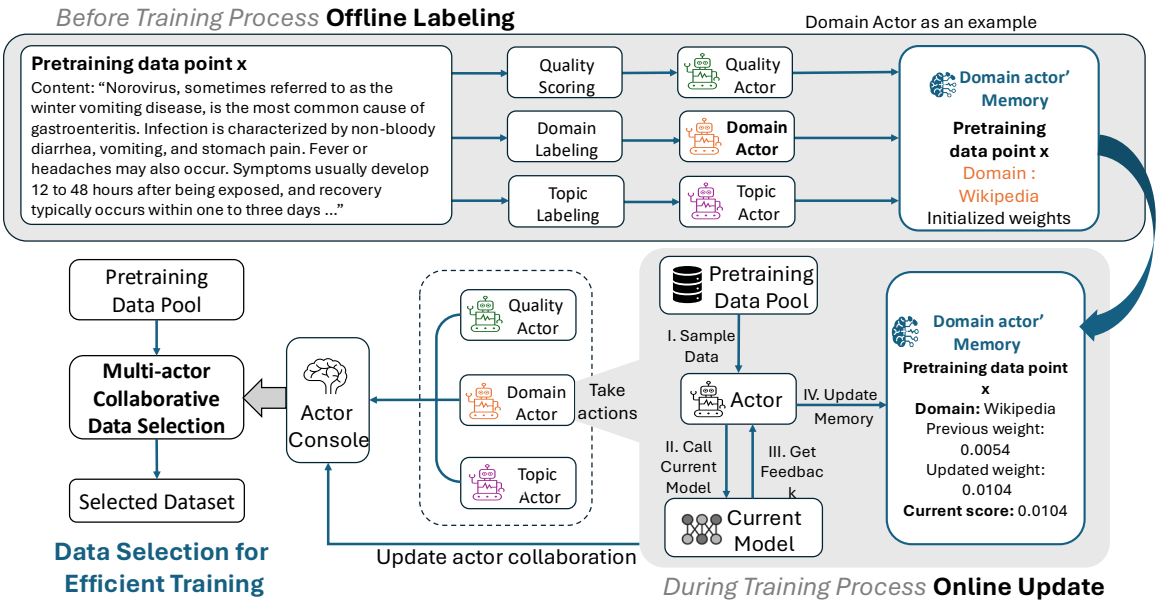


Figure 2: **Illustration of multi-actor collaborative framework.** Multi-actor collaborative framework for pretraining data selection that integrates multiple perspectives by combining offline priors and online model-derived preferences.

3.2 Multi-Actor Data Selection Framework

In order to solve the optimization problem in Equation 2, we develop a framework illustrated in Figure 2. This framework consists of two primary stages: the offline labeling stage and the online update stage. Before the training process, some initial information (i.e., the initialized measurements in some heuristic) is computed for the entire pretraining corpus, and this information is stored separately in each actor's memory (formally defined below). During the training process, the current model (i.e., LM to be trained) is used to update the actors' memory and their collaboration mechanism based on rewards computed on the current model. An actor console is responsible for aggregating the opinions of each actor and making the final data selection decision. Formally, we define the actor in Definition 1 and the actor console in Definition 2. Detailed formulation is in Appendix A.4.

Definition 1 (Actor)**.** An *actor* $\mathcal{A}$ is a data selection method defined by a specific attribute (e.g., quality, domain, or topic) with memory $\mathcal{H}_\mathcal{A}$ that stores labels for each data point and their associated scores. During training, the actor takes several actions: (1) Sample data $\mathcal{D}_\mathcal{A}$ according to predefined sampling distribution, (2) Call the current model to compute the reward $\mathcal{R}(\mathcal{D}_\mathcal{A})$, (3) Get feedbacks from current model state, and (4) Update the internal weights $\mathbf{w}_\mathcal{A}$ in its memory. Then it assigns a score $\mathcal{S}_\mathcal{A}$ to each data point based on its updated memory, prioritizing the good data according to the updated weights. One actor's objective is to maximize its reward by updating this actor's internal weights and increasing the score of higher-reward data points.

Definition 2 (Actor Console)**.** The *actor console* is in charge of coordinate opinions from different actor to make final decision of selecting dataset for next training stage. Specifically, it consolidates scores $\mathcal{S}_\mathcal{A}(x_i)$ from multiple actors $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ to calculate a final score $S(x_i)$ for each data point x_i, and select the final dataset. The console adjusts the collaborative weights $\theta_\mathcal{A}$ for each actor based on their respective aggregate rewards $\mathcal{R}(\mathcal{D}_\mathcal{A})$, balancing their contributions during training. In cases where there are conflicts in the decisions made by actors, the console resolves these by adjusting the weights $\theta_\mathcal{A}$ to prioritize the actors that have a greater positive impact on the model's performance, ensuring an effective data selection process.

Now the reward signal is actually came from multiple actors, the optimization goal in Equation 2 becomes maximizing the expectation of collaborative actors. In our current implementation, we include three actors, which are topic actors, quality actors and domain actors. They are aiming to maximize the rewards from topic, quality and domain perspective respectively. In the following sections, we will detail how we initialize and update a single actor (§3.3), and how we update the actor console for multi-actor collaboration (§3.4).

3.3 Single Actor Initialization and Update

Actor initialization. As defined in Definition 1, for a particular actor, we have to maintain its mem-

ory $\mathcal{H}_{\mathcal{A}}$ throughout the training process. Before training process begin, we label the whole training dataset $\mathcal{D}$ offline and store the labeled information to the memory of corresponding actors. Specifically, for each data point $x_i \in \mathcal{D}$, i.e., a single document in our settings, we first get the quality, topic and domain label using scorer and classifier.

We initialize the weight of the topic actor and domain actor following the RegMix (Liu et al., 2024) framework. Unlike the original RegMix, which only considers mixing data based on domain labels, we examine data mixing weights based on domain as well as the topic labels. We initialize our quality actor similar to the data selection decision of QuRating (Wettig et al., 2024) and FineWeb-Edu (Lozhkov et al., 2024). Further details can be found in §A.7.3. The initial weights for each actor are stored in their respective memory.

During the training phase, we leverage the current model to adjust the weight of each actor. As depicted in Figure 2, at the data selection stage, each actor performs several actions to update its memory and inform decision-making. Take domain actor as an example, it takes three-step action during data selection stage: (1) Sample pretraining data points from the pretraining data pool, distributing them uniformly across each domain; (2) Call the current model to assess the reward of each data point and gather feedback; (3) Update the memory of domain weights based on gathered feedback and adjusts the score for each data point by incorporating prior knowledge from the offline labeling process. This process is similarly followed by the quality actor and the topic actor.

Actor update. After sampling data uniformly from actor search space, each actor updates its internal weights using local information based on the sampled data. For example, for domain actor, it calculates the average influence of each domain. For actor $\mathcal{A} \in \{\mathcal{A}_{\text{Quality}}, \mathcal{A}_{\text{Domain}}, \mathcal{A}_{\text{Topic}}\}$, it updates its internal weights by calculating the overall rewards sampled from each interval as:

$$\mathcal{R}(\mathcal{D}_{\mathcal{A}}^{j}) = \frac{1}{|D_{\mathcal{A}}^{j}|} \sum_{x_i \in D_{\mathcal{A}}^{j}} r(x_i) \coloneqq \overline{R}_{\mathcal{A}}^{j}, \quad (4)$$

where $D_{\mathcal{A}}^{j}$ represents the sample set of the j-th subcategory under actor $\mathcal{A}$, e.g. Wikipedia for domain actor. And x_i is a sample within this sample set. The sliding averaging is used to update the weight for each subcategory $w_{\mathcal{A}}^{j}$ with current rewards:

$$w_{\mathcal{A}}^{j} \leftarrow (1 - \eta_A) \cdot w_{\mathcal{A}}^{j} + \eta_A \cdot \overline{R}_{\mathcal{A}}^{j}, \quad (5)$$

where $\eta_{\mathcal{A}}$ is the sliding average factor to trades-off bias-variance. The overall updated weight of actor $\mathcal{A}$ is an vector in $n_{\mathcal{A}}$ dimension, $\mathbf{w}_{\mathcal{A}} = [w_{\mathcal{A}}^{1}, ..., w_{\mathcal{A}}^{n_{\mathcal{A}}}]$, where $n_{\mathcal{A}}$ is the number of total subcategory within the space of actor $\mathcal{A}$. Utilizing the prior memory stored by each actor, it can give out a final score for each data point as $\mathcal{S}_{\mathcal{A}}(x_i) = w_{\mathcal{A}}^{j}$, where j is the subcategory that x_i belongs to.

3.4 Multi-Actor Collaboration

Ultimately, the actor console defined in Definition 2 aggregates all actors' feedback to compute a final score for each data point, determining the final data selection decision.

Multi-actor collaboration. In the context of multi-actor collaboration, the weighted score for each actor must be calculated to evaluate their respective contributions effectively. This calculation takes into account various factors specific to each actor. For every data sample x_i, the overall score $S(x_i)$ is determined by the following formula:

$$\begin{aligned} S(x_i) = (&\theta_{\text{Quality}} \cdot \mathcal{S}_{\text{Quality}}(x_i) + \\ &\theta_{\text{Domain}} \cdot \mathcal{S}_{\text{Domain}}(x_i) \ + \theta_{\text{Topic}} \cdot \mathcal{S}_{\text{Topic}}(x_i)), \end{aligned} \quad (6)$$

where $\mathcal{S}_{\text{Quality}}(x_i)$, $\mathcal{S}_{\text{Domain}}(x_i)$, and $\mathcal{S}_{\text{Topic}}(x_i)$ are scores calculated by quality, domain, and topic actors for sample x_i. $\theta_{\mathcal{A}} \in \{\theta_{\text{Quality}}, \theta_{\text{Domain}}, \theta_{\text{Topic}}\}$ is the collaborative weight for each actor, which is updated during training process.

Collaborative weight update. To dynamically adjust the importance of each actor during various training phases, we modify the actor's collaborative weight based on its overall rewards. We compute the reward of each actor and the average reward across all actors:

$$\overline{R}_{\mathcal{A}} = \frac{1}{|n|} \sum_{j=1}^{n} w_{\mathcal{A}}^{j} \cdot \overline{R}_{\mathcal{A}}^{j}, \quad \overline{R} = \frac{1}{3} \sum_{\mathcal{A}} \overline{R}_{\mathcal{A}}, \quad (7)$$

This information is then used to update each actor's collaborative weight, which is stored in the actor console's memory for future decision-making:

$$\theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} + \eta_{\mathcal{A}} \cdot (\overline{R}_{\mathcal{A}} - \overline{R}). \quad (8)$$

By continuously refining these weights, the collaboration strategy adapts to optimize overall performance and appropriately adjust the role of each actor throughout different stages of training. Complete training pipeline is outlined in Algorithm 2.

Table 1: Our approach boosts model performance across tasks. To fit demonstrations within 1024 tokens, we provide full results for 0, 3, and 5 shots in Appendix A.8. The table covers Problem Solving, Commonsense Reasoning, and Reading Comprehension, with best QuRating and DSIR variants: QuRating-Edu and DSIR-Wiki.

Selection Method	Problem Solving *(4 tasks)*	Commonsense Reasoning *(4 tasks)*	Reading Comprehension *(2 tasks)*	Average *(10 tasks)*
Random sampling - 30B tokens	31.1	32.9	43.1	34.2
Random sampling - 60B tokens	33.6$^{\uparrow 2.5}$	33.7$^{\uparrow 0.8}$	46.1$^{\uparrow 3.0}$	36.1$^{\uparrow 1.9}$
Perplexity PPL (Ankner et al., 2024)	29.9$^{\downarrow 1.2}$	30.5$^{\downarrow 2.4}$	42.4$^{\downarrow 0.7}$	32.7$^{\downarrow 1.5}$
Classifier-based data selection				
QuRating (Wettig et al., 2024)	34.1$^{\uparrow 3.0}$	34.1$^{\uparrow 1.2}$	41.4$^{\downarrow 1.7}$	35.6$^{\uparrow 1.4}$
FineWeb-Edu (Penedo et al., 2024)	32.6$^{\uparrow 1.5}$	33.0$^{\uparrow 0.1}$	45.3$^{\uparrow 2.2}$	35.3$^{\uparrow 1.1}$
DSIR (Xie et al., 2023b)	30.9$^{\downarrow 0.2}$	32.0$^{\downarrow 0.8}$	41.5$^{\downarrow 1.6}$	33.5$^{\downarrow 0.7}$
Domain mixing methods				
DOGE (Fan et al., 2024)	30.9$^{\downarrow 0.2}$	32.2$^{\downarrow 0.7}$	45.1$^{\uparrow 2.0}$	34.3$^{\uparrow 0.1}$
DoReMi (Xie et al., 2023a)	30.4$^{\downarrow 0.7}$	32.6$^{\downarrow 0.3}$	44.8$^{\uparrow 1.7}$	34.1$^{\downarrow 0.1}$
DMLaw (Ye et al., 2024)	30.2$^{\downarrow 0.9}$	32.1$^{\downarrow 0.9}$	45.1$^{\uparrow 2.0}$	33.9$^{\downarrow 0.3}$
RegMix (Liu et al., 2024)	30.7$^{\downarrow 0.4}$	32.5$^{\downarrow 0.4}$	44.6$^{\uparrow 1.5}$	34.2$^{\uparrow 0.0}$
Influence MATES (Yu et al., 2024)	30.9$^{\downarrow 0.2}$	34.0$^{\uparrow 1.1}$	46.5$^{\uparrow 3.4}$	35.3$^{\uparrow 1.1}$
Multi-actor collaboration (ours)	36.7$^{\uparrow 5.6}$	34.8$^{\uparrow 1.9}$	45.9$^{\uparrow 2.8}$	37.8$^{\uparrow 3.6}$

4 Experiments

We conduct a series of experiments to evaluate the effectiveness of our multi-actor collaborative data selection method. Comprehensively, we find that: (1) In the end-to-end experiments, our approach introduces significant improvement in terms of data efficiency leading to faster convergence for LM training, and achieves up to 10.5% improvements on average across various language model benchmarks when compared with other baseline approaches (§4.1); (2) We also verify that the design and implementation of the core components in our multi-actor framework design are necessary to reach this advanced performance through a set of carefully designed ablation studies (§4.2).

4.1 End-to-end Experiments

We evaluate our multi-actor framework against a wide category of state-of-the-art approaches to compare the data efficiency for LM pretraining. We train a 1.3 billion parameter LLAMA-2 architecture model with 30 billion selected tokens. We also report our results of generalizing our methods to **3.6B** and **8B** models in Appendix A.1.

Experimental setup. We first enumerate the experimental setup as below:

- **Pretraining datasets.** We utilize the popular SlimPajama (Soboleva et al., 2023) dataset including 627 billion tokens, which is derived from the RedPajama (Computer, 2023) dataset. The SlimPajama (Soboleva et al., 2023) provide the meta-data about the domain information for each sample. Before the training process, we annotate the entire dataset using the FineWeb-Edu quality scorer (Penedo et al., 2024) along with our custom-trained BERT-based topic classifier. The training details for the topic classifier is provided in Appendix §A.2.
- **Training details.** We adopt the model architecture from LLAMA-2 (Touvron et al., 2023b) at the scale of 1.3 billion parameters (see the detailed configuration in Appendix §A.7-Table 9). Following the principles of the scaling law (Hoffmann et al., 2022) and the DCLM framework (Li et al., 2024), we decide to use a total of 30 billion tokens. All training tokens are sampled from the 670 billion-token SlimPajama (Soboleva et al., 2023) dataset using various sampling strategies. Further details regarding the training process can be found in §A.7.
- **Evaluation benchmarks.** To evaluate the pretrained models thoroughly, we conduct extensive assessments across various downstream tasks, categorized into three areas: (1) problem solving: MMLU (Hendrycks et al., 2021), ARC-Easy/Challenge (Clark et al., 2018), and MathQA (Welbl et al., 2017); (2) commonsense reasoning: SIQA (Sap et al., 2019),

WinoGrande (Sakaguchi et al., 2020), OpenbookQA (Mihaylov et al., 2018), and CommonsenseQA (Talmor et al., 2019); (3) reading comprehension: RACE (Lai et al., 2017) and BoolQ (Clark et al., 2019). Evaluations are conducted using the `lm-evaluation-harness` framework (Gao et al., 2023) in an in-context learning setting, and average accuracy is reported for easy comparison.

- **Baselines.** We select a wide range of baselines to conduct extensive the data efficiency comparison, where these methods can be classified to five main categories: (1) *random sampling*, we test this policy with both the standard data volume of 30B tokens and a supplemented version with 60B tokens; (2) *perplexity*-based data selection (Ankner et al., 2024); (3) *classifier*-based data selection, where we select the following methods: QuRating (Wettig et al., 2024), FineWeb-Edu (Penedo et al., 2024), DSIR-Book (Xie et al., 2023b) and DSIR-Wiki (Xie et al., 2023b); (4) *domain mixing*-based methods, where we select the following methods: DOGE (Fan et al., 2024), DoReMi (Xie et al., 2023a), DMLaw (Ye et al., 2024) and RegMix (Liu et al., 2024); and (5) *influence function* based methods for online data selection, i.e., MATES (Yu et al., 2024). Implementation details of these baselines can be found in §A.7.2.

Results. We present the results of three types of downstream tasks in Table 1, with the complete 0-shot (Table 11), 3-shot (Table 12), and 5-shot (Table 13) results for all tasks enumerated in §A.8. We highlight that *our methods show a substantial improvement in the average performance across all downstream tasks when compared with all the baselines*. Concretely, we observe that when compared with the *random sampling* based approach, our method not only significantly outperforms the standard 30 billion token setup but also surpasses the model trained on 60 billion tokens with a performance gain of 4.7%. Similarly, we also show an improvement of 15.6% compared with *perplexity*-based data selection (Ankner et al., 2024), an improvement of up to 6.2% compared with *classifier*-based data selection, an improvement of up to 10.2% compared with *domain mixing*-based methods, and an improvement of 7.1% compared with *influence function* based approach, i.e., MATES (Yu et al., 2024).

Discussion. We highlight that our proposed multi-actor collaborative data selection mechanism introduces statistical efficiency in terms of LM training convergence and also provides some computational efficiency in terms of data processing overheads. In terms of *statistical efficiency*, our method consistently outperforms others at every benchmarked training step, as shown in Figure 3. While MATES (Yu et al., 2024) performs comparably to our methods during the early training phase (steps 1500 to 3000), its performance drops in later stages. This aligns with its original paper, which notes that relying solely on influence functions for specific reference tasks (e.g., LAMBADA (Paperno et al., 2016)) can degrade performance in mid-to-late pretraining. Despite this, MATES still outperforms other methods without dynamic adjustments shown in Figure 3. In contrast, our multi-actor collaborative data selection mechanism can dynamically adjust the corresponding weights from different actors and select data based on the most up-to-date model preferences, effectively mitigating biases and surpassing other domain-mixing and data-selection techniques. In terms of *computational efficiency*, we also achieve higher computational efficiency than previous methods. For example, QuRating (Wettig et al., 2024) requires around 7.13×10^{20} FLOPs to label the entire SlimPajama dataset, while our offline labeling takes just 9.91×10^{19} FLOPs. MATES (Yu et al., 2024), which recalculates influence scores and trains a BERT model for each labeling cycle, incurs 1.98×10^{20} FLOPs for a four-stage update. Additionally, MATES' labels are only usable in the next training stage, making it time-consuming and difficult to scale. In contrast, our method can improve the computational efficiency from two aspects: (1) we find that a group of light-weight actors collaboratively enables superior data selection, which is more computational efficiency than any method that requires a heavy data processing or label procedure; (2) the collaborative, dynamic learning procedure introduced in our multi-actor framework is computational efficient; by using a sampled holdout set and CPU-based calculations for updating actor parameters, our computational overhead is ignorable compared with heavy LM training computation.

4.2 Ablation Study

We introduce a set of carefully designed ablation studies to justify the design and implementation of our multi-actor collaborative data selection frame-

Table 2: This ablation study examines the performance of various combinations of actor collaboration and update mechanisms. All models are in 1.3B LLaMA2 architecture. Three-shot accuracy is reported for all tasks, with the highest value in each column shown in **bold**.

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Quality&Domain&Topic actor	**65.8**	**31.5**	**23.0**	26.6	**24.6**	39.9	**54.1**	20.1	**60.4**	30.5	**37.7**
without collaboration update	59.4	26.3	21.3	25.1	20.5	38.9	52.9	19.8	58.1	28.3	35.1
Domain&Quality actor	63.3	29.7	22.6	25.1	21.8	**40.5**	53.1	20.3	59.5	28.8	36.5
Topic&Quality actor	62.9	28.1	22.3	26.5	22.6	39.6	51.8	**21.7**	56.7	**30.7**	36.3
Domain&Topic actor	55.6	25.2	21.8	26.5	23.1	39.1	53.7	20.9	57.5	29.0	35.2
Quality actor	59.1	29.7	22.4	25.3	21.1	38.5	51.2	19.1	57.2	28.3	35.2
Domain actor	54.1	25.6	21.4	25.9	22.3	38.1	53.6	20.0	58.1	27.9	34.7
Topic actor	55.3	25.3	21.9	**27.1**	22.1	39.4	51.5	19.8	56.3	28.9	34.8
No actor	54.6	23.0	22.1	24.9	18.8	40.3	52.9	21.5	53.0	29.8	34.1

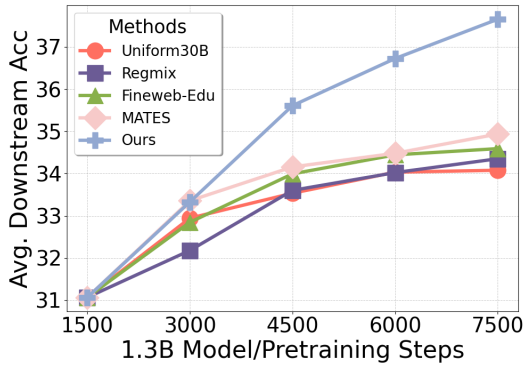


Figure 3: Downstream three-shot performance of the 1.3B model in relation to pretraining steps, using 7500 steps for 30B tokens. Our methods outperform baselines from all categories.

work. Concretely, (1) we test the combination of different actors to show the advance introduced by collaboration, and (2) we verify the necessity of the dynamic adjustment of the actor's weight for data selection.

Results and discussion. The results of the ablation study are shown in Table 2. We want to highlight the result from two aspects: First, the ablation study underscores *the importance of each actor in achieving optimal performance* across the training tasks. When the quality, domain, and topic actors are used together, the model performs best, highlighting the benefits of their combined use, as shown in Table 2. In terms of evaluating *each actors' contributions*, we find that the quality actor excels in problem-solving tasks like ARC-E and MathQA by leveraging educational knowledge but is less effective for domain-specific or context-heavy tasks like BoolQ and RACE. The domain actor enhances commonsense reasoning (e.g., CommonsenseQA) and reading comprehension (e.g., BoolQ) by incorporating domain-specific knowledge. The topic actor is most effective for multi-topic tasks like MMLU and contributes significantly to commonsense reasoning tasks like SocialIQA. Second, the ablation study verifies the design and implementation of *the collaborative dynamic adjustment of the actors' weights* (introduced §3.4) for efficient data selection. When actors were initialized with equal, fixed weights instead of using dynamic weighting, overall performance dropped significantly, as shown in Table 2.

5 Related Work

Data selection in LM pretraining. Selecting high-quality pretraining data from large corpora is crucial for effective LM training. Recent approaches leverage various methodologies for efficient data selection. Concretely, *classifiers* (Brown et al., 2020; Chowdhery et al., 2023; Du et al., 2022; Xie et al., 2023b) and *language modeling perplexity* (Wenzek et al., 2020; Thrush et al., 2024) have been applied to identify data resembling high-quality samples; recently, more advanced quality scores based on classifier have shown the effectiveness in data selection, e.g., QuRating (Wettig et al., 2024), FineWeb-Edu (Lozhkov et al., 2024), etc. *Data mixture* is another effective way to improve data diversity, at both token level (Touvron et al., 2023a; Gao et al., 2020; Soboleva et al., 2023) and sample level, e.g., DoReMi (Xie et al., 2023a), DOGE (Fan et al., 2024), DMLaw (Ye et al., 2024), and RegMix (Liu et al., 2024); very recently, topic distributions has also been considered as an effective data mixing method, e.g., the downsampling over-represented topics in Llama 3.1 (Team, 2024). *Influence functions* have been studied to understand for data efficiency (Koh and Liang, 2017), and some recent attempts based on efficient approxima-

tion have been proposed to improve data efficiency in LM pretraining (Schioppa et al., 2022; Grosse et al., 2023; Isonuma and Titov, 2024); For example, MATES (Yu et al., 2024) uses a staged BERT model to assess data influence, QUAD (Zhang et al., 2024) leverages cluster information to reduce the computational cost of calculating individual data influence.

Multi-agent and multi-actor collaborative frameworks. Collaboration across multiple autonomous or semi-autonomous entities has been studied extensively under the paradigms of multi-agent and multi-actor systems. Traditional multi-agent systems (Russell and Norvig, 2016; Wooldridge, 2009) involve autonomous agents that make independent decisions and learn through interaction, often coordinated via mechanisms such as reward signals, negotiation, or centralized planning. These systems have been effective in applications such as neural architecture search (Bello et al., 2017), collaborative large language model programming (Hong et al., 2024), and distributed control (Olfati-Saber, 2006). In contrast, our work adopts a multi-actor perspective, where actors are not general-purpose intelligent agents, but specialized components—specifically, independently operating data selection methods. Each actor follows a distinct heuristic to prioritize pretraining data, adapting dynamically to the model's evolving state. A central coordination mechanism integrates these priorities over time, resolving conflicts and guiding collective behavior. This aligns with formalizations of multiactor systems, where the term "actor" serves as a generic abstraction that encompasses agents, bodies, or effectors (Russell and Norvig, 2016). Unlike multi-agent systems, which may involve deliberative autonomy and decentralized goal negotiation, multiactor systems often assume a shared objective and focus on concurrent action, coordination, and conflict resolution. Our framework draws from both traditions: it retains the modularity and adaptability of multi-agent systems, while leveraging the structured concurrency and coordination principles emphasized in multiactor planning (Ligtenberg et al., 2001, 2004; Wang et al., 2020). This hybrid approach enables us to tackle a largely underexplored challenge: resolving competition among heterogeneous data selection heuristics to improve the efficiency and effectiveness of language model pretraining.

6 Conclusions

In this paper, we introduce a multi-actor collaborative data selection framework to enhance efficiency in LM pretraining. Our framework allows multiple data selection methods to operate as independent actors, with an actor console designed to dynamically integrate their outputs throughout the LM training process. Empirical studies show it improves data efficiency, speeds up convergence, and achieves up to 10.5% average performance gains over state-of-the-art methods. These results demonstrate the effectiveness of dynamically combining data selection strategies to resolve conflicts and optimize LM pretraining.

Limitations

While our method greatly improves data selection for language model pre-training, our study has some limitations. Due to computational constraints, our experiments were limited to relatively small-scale models (up to 8B parameters) with restricted token budgets. Additionally, while our quality metrics are comprehensive, they may not fully capture all dimensions of pre-training data. In future work, we plan to refine or expand these metrics to bridge these gaps.

Acknowledgement

This work is supported by the HKUST startup grant R9895 from CSE, RGC-ECS project 26218024, Shanghai Artificial Intelligence Laboratory, the National Key R&D Program of China (2022ZD0160201), the National Key R&D Program of China (2024YFA1014003), National Natural Science Foundation of China (92470121, 62402016), and CAAI-Ant Group Research Fund. We express sincere thanks to InternTrain Team of Shanghai Artificial Intelligence Laboratory, especially Yang Gao, for their kind help for the pretraining experiments.

References

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. 2024. Perplexed by perplexity: Perplexity-based pruning with small reference models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. 2017. Neural optimizer search with reinforcement learning. In *International Conference on Machine Learning*, pages 459–468. PMLR.

Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

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

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. 2024. Dsdm: Model-aware dataset selection with datamodels. In *International Conference on Machine Learning*, pages 12491–12526. PMLR.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. 2024. DOGE: Domain reweighting with generalization estimation. In *Forty-first International Conference on Machine Learning*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. 2023. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *ICLR*.

Masaru Isonuma and Ivan Titov. 2024. Unlearning traces the influential training data of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6312–6325.

Carolyn Kim, Ashish Sabharwal, and Stefano Ermon. 2016. Exact sampling with integer linear programs and random perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*.

Arend Ligtenberg, Arnold K Bregt, and Ron Van Lammeren. 2001. Multi-actor-based land use modelling: spatial planning using agents. *Landscape and urban planning*, 56(1-2):21–33.

Arend Ligtenberg, Monica Wachowicz, Arnold K Bregt, Adrie Beulens, and Dirk L Kettenis. 2004. A design and application of a multi-agent system for simulation of multi-actor spatial planning. *Journal of environmental management*, 72(1-2):43–55.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. 2024. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu.

Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Reza Olfati-Saber. 2006. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420.

OpenAI. 2024. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`.

Denis Paperno, German David Kruszewski Martel, Angeliki Lazaridou, Ngoc Pham Quan, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda Torrent, Fernández Raquel, et al. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *The 54th Annual Meeting of the Association for Computational Linguistics Proceedings of the Conference: Vol. 1 Long Papers*, volume 3, pages 1525–1534. ACL.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Common Crawl Project. 2007. Common crawl: Open repository of web crawl data.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. Pearson.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Llama Team. 2024. The Llama 3 Herd of Models.

Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. 2024. Improving pretraining data using perplexity correlations. *arXiv preprint arXiv:2409.05816*.

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

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

Bojan Tunguz. 2019. Jeopardy! questions.

Ni Wang, Petra W Heijnen, and Pieter J Imhof. 2020. A multi-actor perspective on multi-objective regional energy system planning. *Energy Policy*, 143:111578.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. Qurating: Selecting high-quality data for training language models. In *Forty-first International Conference on Machine Learning*.

Michael Wooldridge. 2009. *An introduction to multiagent systems*. John wiley & sons.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning*.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023a. Doremi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, pages 69798–69818. Curran Associates, Inc.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. 2023b. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems*, volume 36, pages 34201–34227. Curran Associates, Inc.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. 2024. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*.

Zichun Yu, Spandan Das, and Chenyan Xiong. 2024. Mates: Model-aware data selection for efficient pretraining with data influence models. *Advances in Neural Information Processing Systems*, 37:108735–108759.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ye Yuan, Guoren Wang, and Conghui He. 2024. Harnessing diversity for important data selection in pretraining large language models. *Preprint*, arXiv:2409.16986.

A Appendix

A.1 Generalization to 3.6B and 8B Models

To evaluate the scalability of our approach, we conducted additional experiments training a 3.6 billion parameter model based on the LLaMA 3.2 architecture, which further demonstrates the scalability of our method. So far we have trained on 36 billion tokens and achieved strong performance, with plans to continue training with additional tokens according to scaling laws. As Table 3 shows, when compared to random selection, our method shows consistent performance improvements across all downstream tasks, achieving a 13.7% increase in average accuracy—significantly higher than the 10.5% improvement observed with the 1.3B models. Our results in Table 4 show that our methods outperform random selection approaches by 10.2% for 8B LLaMA 3.2 architecture models. Based on trends across three different model sizes (373M, 1.3B, 3.6B and 8B), our approach consistently outperforms random selection by over 10% on average. This consistent advantage makes us believe that it suggests our method has strong potential for training even larger models, including those with 10B+ parameters.

A.2 Details of Training Topic Classifier

As shown in Figure 4, we first cluster 1.4 billion documents obtained from Common Crawl (Project, 2007) into 10,000 clusters using KNN. And we use GPT-4o (OpenAI, 2024) to generate a summary for the content in each cluster. Additionally, we implement two parallel steps: unsupervised and supervised. In the unsupervised step, we perform secondary clustering of the 10,000 clusters into 100 clusters, from which we extract 20 summaries for each cluster. We utilize GPT-4o to extract category labels, refining these into a coherent hierarchical labeling system for the classification of 42 distinct topics.

In the supervised data processing step, leveraging Gopher cleaning ruls (Rae et al., 2021) and Min-Hash (Broder, 1997) deduplication, we clean the whole datasets and cluster the datasets into 10,000 clusters. We then extract 50 equidistant samples from each cluster. This process yields approximately 500,000 data points, which we categorize into the aforementioned 42 topics by calling GPT-4o (OpenAI, 2024) using the prompt shown below:

Since GPT-4o is not specialized for classification tasks, we obtain actual topic data with slightly more than 42 topics, as shown in Figure 6. We then manually summarize the topics provided by GPT-4 into 13 categories, ensuring that the subtopics within each category shared similarities. The detailed category distributions appear in Figure 6, along with specific clustering information. Ultimately, we employ the annotated data to fine-tune a BERT-like regression model (Devlin et al., 2018). Following model classification, we conduct human proofreading to ensure accuracy, and we present the final results below.

A.3 Guidelines for Generalizing a New Criteria as an actor

This section provides a detailed guidelines for incorporating a new criterion into our multi-actor system. The process is designed to ensure seamless integration and effective collaboration between existing and new actors.

Our framework offers several significant benefits when integrating a new actor. First, it offers **flexibility**, as the addition of new criteria can be performed independently of the core framework. This decoupling ensures that introducing new components does not require significant structural changes, allowing for smooth integration with minimal disruption to existing processes. Second, the approach is highly **scalable**. By enabling the training of new classifiers offline, the system can be easily adapted to handle a wide variety of data selection goals, which can evolve over time as new criteria emerge. Finally, the framework ensures **extensibility**, meaning it can seamlessly accommodate new objectives, whether simple or complex. This extensibility is key to maintaining efficient and effective collaboration across multiple actors, regardless of the size or complexity of the task at hand.

As demonstrated in Algorithm 2, our framework seamlessly integrates a new actor through a series of straightforward steps. This approach ensures that the multi-actor framework remains flexible and effective in addressing diverse data selection objectives while preserving its collaborative efficiency.

A.4 Connection of Multi-actor Collaborative Selection Method to Multi-agent RL

The proposed multi-actor collaborative selection method is fundamentally inspired by the intelligent actor defined in (Russell and Norvig, 2016), where the actor generally refers to an entity that perceives some status and map the observed sta-

Table 3: This study compares the performance of training **3B model** from scratch on 60B tokens selected using random sampling versus multi-actor collaboration. Accuracy is reported for all tasks, with the highest value in each column shown in **bold**.

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Random Sampling	34.8	17.7	21.3	23.0	12.0	32.9	50.2	19.6	37.8	20.9	27.0
Multi-actor collaboration (Ours)	**42.9**	**21.3**	**21.9**	**24.0**	**15.8**	**33.9**	**51.0**	**20.4**	**54.8**	**21.2**	**30.7**

Table 4: This study compares the performance of training **8B model** from scratch on 25B tokens selected using random sampling versus multi-actor collaboration. Accuracy is reported for all tasks, with the highest value in each column shown in **bold**.

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Random Sampling	22.2	53.3	21.5	23.3	19.4	36.1	51.0	18.2	48.0	21.3	31.4
Multi-actor collaboration (Ours)	**25.5**	**58.0**	**23.2**	**24.1**	**21.6**	**38.8**	**53.1**	**20.8**	**57.8**	**22.9**	**34.6**

tus into actions. However, our framework also has many similarity compare with traditional multi-agent framework in reinforcement learning, where multiple actors work together to optimize a shared objective. In this section, we formally demonstrate the relationship between our framework and traditional multi-agent RL.

A.4.1 Overall Definition in Reinforcement Learning Formulation

We first clearly formulate each component of framework compare with components in general MARL framework for understanding the mechanism of our framework. As our goal is to select the opt the **global action** at each step involves selecting a subset of data, $\mathcal{D}_k$, from the entire dataset, $\mathcal{D}$. This subset is used to update the model, where batches are drawn from $\mathcal{D}_k$ for training. The **global state** is represented by the current model parameters, M, which evolve as the model is trained. The **state transition** is formalized as $M' = \mathcal{O}(M, \mathcal{D}_k)$, where M' denotes the updated model after training on $\mathcal{D}_k$.

The **reward function** measures the improvement in model performance on a reference task, $\mathcal{T}_{\text{ref}}$, which serves as a proxy for the true downstream task $\mathcal{T}_{\text{eval}}$. The reward is defined as:

$$R(M'|M, \mathcal{T}_{\text{ref}}) = \mathbb{E}_{x\sim\mathcal{T}_{\text{ref}}}[-l(x; M')],$$

where $l(x; M')$ is the loss on $\mathcal{T}_{\text{ref}}$. For individual data points, the reward is estimated using influence functions:

$$r(x_i) = \text{Influence}(x_i, M, \mathcal{T}_{\text{ref}}).$$

This formulation links data selection directly to its impact on improving the model's performance on $\mathcal{T}_{\text{ref}}$.

A.4.2 Actor Design

The estimation of value is as follows: The actor stores a reward estimation vector for each subset. The update rule is given by

$$w^j_{\mathcal{A}}(t+1) = (1-\eta_{\mathcal{A}}) \cdot w^j_{\mathcal{A}}(t) + \eta_{\mathcal{A}} \cdot \bar{R}^j_{\mathcal{A}}.$$

The sliding average is used here because if all data in a subset were fully processed to compute $\bar{R}^j_{\mathcal{A}}$, there would be no need for a sliding average. However, since only a portion of the data is sampled, the estimate has higher variance, which is not favorable for training. At the same time, the influence score itself is dynamic (even if the data remains constant, the model evolves). Averaging with outdated scores introduces bias. Therefore, the sliding average factor $\eta_{\mathcal{A}}$ strikes a 'bias-variance tradeoff'. We assume the score estimate for each data point x_i in $\mathcal{D}^j_{\mathcal{A}}$ with respect to the dimension of interest for the actor, is given by $S_{\mathcal{A}}(x_i) = w^j_{\mathcal{A}}$ where $x_i \in \mathcal{D}^j_{\mathcal{A}}$.

A.4.3 Multi-actor Collaboration

Assume that the score of a single data point in the reference task is obtained as a weighted sum of multiple components. The total score for each data point is given by Equation 7 as:

$$S(x_i) = \sum_{\mathcal{A}} \theta_{\mathcal{A}} \cdot S_{\mathcal{A}}(x_i),$$

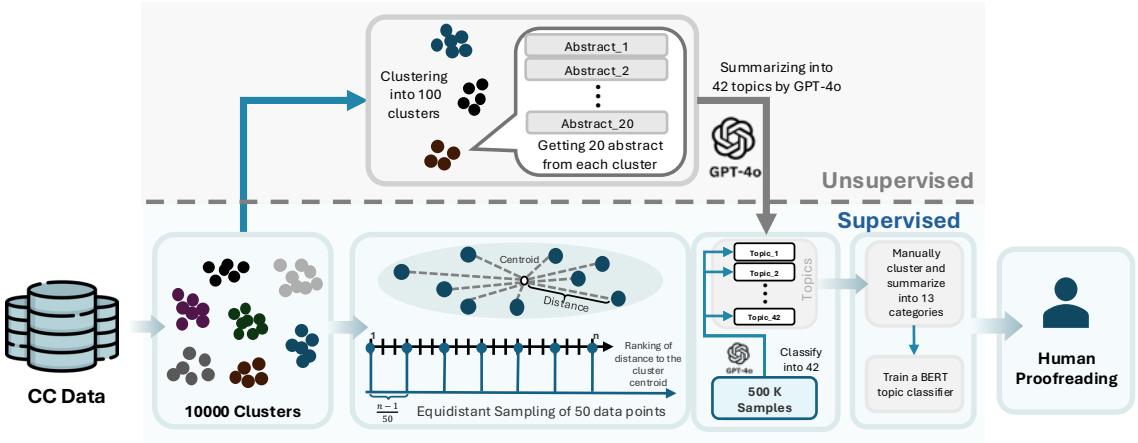


Figure 4: **Illustration of training process for topic classifier.** This diagram shows the process of training a BERT-based topic classifier using CommonCrawl data. 1.44 billion documents are clustered to generate topics. GPT-4o handles topic summarization and annotation, while a BERT model is trained to classify 13 topics, with humans doing final proofreading.

Prompt Template

Task: You are an annotator responsible for analyzing the category distribution of web data. For each provided full-text segment, select one topic from the following list of 42 that best represents the primary focus of the text:

Topic: [Animals, Culture, Travel, Politics, Art, Architecture, Consumerism, Social Networking, Investment, Environment, Industry, Technology, Entertainment, Hobbies, Music, History, Healthcare, Fashion, Beauty, Relationships, Real Estate, Careers, Agriculture, Local, Harmful information, Adult, Sports, News, Pets, Community, Home, Health, Business, Literature, Food Culture, Transportation, Law, Science and Technology, Finance, Education, Automotive, Society]

Response Format: Please respond with a JSON object structured as follows:
{
"selected topic": string //selected topic from the 42 topics
"explanation": string //explain why reorganize the topic in this way
}

Figure 5: We illustrate the prompt construction process for GPT-4 to reorganize the topic of 500k data points.

where $\theta_{\mathcal{A}}$ are collaborative weights. A central coordinator adjusts these weights over time based on the actors' contributions to the overall reward:

$$\theta_{\mathcal{A}}(t+1) = \theta_{\mathcal{A}}(t) + \eta_{\mathcal{A}}(\bar{R}_{\mathcal{A}} - \bar{R}),$$

where $\bar{R}_{\mathcal{A}}$ is the actor's average reward, and $\bar{R}$ is the global average reward:

$$\bar{R}_{\mathcal{A}} = \frac{1}{n}\sum_{j=1}^{n} w_{\mathcal{A}}^{j} \cdot \bar{R}_{\mathcal{A}}^{j}, \quad \bar{R} = \frac{1}{3}\sum_{\mathcal{A}} \bar{R}_{\mathcal{A}}.$$

We consider three possible cases for our framework, comparing its relationship with traditional optimization problem.

- **Single-actor case**: If only one actor is involved, θ becomes irrelevant, reducing the problem to a classical optimization scenario where the actor greedily selects the optimal data based on one criteria.

- **Multi-actor competitive mechanism**: When multiple actors are present, θ reflects each actor's capability. Selecting the best-performing actor for decision-making introduces a heuristic competitive mechanism, building upon the classical optimization framework.

- **Multi-actor collaborative mechanism**: Alter-

Algorithm 1 Integrating a New Criterion into Multi-Actor Collaboration

Require: Sampled dataset $\mathcal{D}_{\text{sample}}$, pretraining dataset $\mathcal{D}_{\text{train}}$, reference dataset D_{ref}, existing Actors $\{\mathcal{A}_i\}_{i=1}^{3}$, scoring weights $\{\theta_i\}_{i=1}^{|\mathcal{D}|}$, memory $\mathcal{H}_{\mathcal{A}}$ for each Actor.

Annotating Data for the New Criteria
  Sample data from the whole datasets, and annotate sampled dataset $\mathcal{D}_{\text{sample}}$ according to new criterion.
Training a Classifier for the New Criteria
  Train a supervised classifier on $\mathcal{D}_{\text{sample}}$.
Defining the New Actor
- **Action Space:** Sample and assess data points across subcategories of the new Criteria.
- **Memory:** Store prior scores and update based on model feedbacks.

Labeling the Pretraining Dataset
  Use trained classifier to label the whole training dataset $\mathcal{D}_{\text{train}}$ and store these labels in memory $\mathcal{H}_{\mathcal{A}_{\text{new}}}$.
Defining Actor Weights and Collaboration Strategy
  1. Define the subcategory weights updating mechanism for $\mathcal{A}_{\text{new}}$ using Eq. 5:

$$w_{\text{new}}^{j} \leftarrow (1-\eta_{\text{new}}) \cdot w_{\text{new}}^{j} + \eta_{\text{new}} \cdot R_{\text{new}}^{j}.$$

  2. Integrate $\mathcal{A}_{\text{new}}$ into the scoring function using Eq. 6:

$$S(x_i) = \theta_{\text{Quality}} S_{\text{Quality}}(x_i) + \theta_{\text{Domain}} S_{\text{Domain}}(x_i) + \theta_{\text{Topic}} S_{\text{Topic}}(x_i) + \theta_{\text{new}} S_{\text{new}}(x_i).$$

Actor Initialization with Regression Techniques
  Initialize the Actor weights w_{new} using regression techniques (e.g., `RegMix`).

Algorithm 2 Multi-actor collaborative data selection for LM pretraining

Require: Training data $\mathcal{D}$, reference task $\mathcal{D}_{\text{ref}}$, main model $\mathcal{M}$, optimizer $\mathcal{O}$, total training steps T, selected size k, update step U, Memory for each actor $\mathcal{H}_{\mathcal{A}}$

Initialize model parameters for main model $\mathcal{M}$
Initialize $\mathcal{D}_k$ as a size-k randomly sampled subset from D
for $t = 1$ to T **do**
  if $t \bmod U = 0$ **then**
    for each actor $\mathcal{A}$ **do**
      Sample data points according to actor's predefined sampling distribution
      Compute rewards $\mathcal{R}_{\mathcal{M}}(x_i; \mathcal{D}_{\text{ref}})$ for each sampled data point x_i
      Update actor weight $\mathbf{w}_{\mathcal{A}} \leftarrow \mathbf{w}_{\mathcal{A}} + \eta_{\mathcal{A}} \cdot \overline{\mathbf{R}}_{\mathcal{A}}$
    end for
    Compute actor score $\overline{R}_{\mathcal{A}}$ and average score $\overline{R}$ according to Eq. 7
    Update collaborative weight $\theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} + \eta_{\mathcal{A}} \cdot (\overline{R}_{\mathcal{A}} - \overline{R})$.
    Calculate coordinator score $S(x_i)$ for $x_i \in \mathcal{D}$ according to Eq. 6
    Select dataset for next training stage $\mathcal{D}_k \leftarrow \text{Top-}k(S(x_i))$ for $x_i \in \mathcal{D}$
  end if
  Sample a batch of data B from $\mathcal{D}_k$
  Update Main Model $\mathcal{M} \leftarrow \mathcal{O}(\mathcal{M}, B)$
end for

natively, when multiple actors are involved, θ can be used to weigh each actor's contributions for decision-making. This introduces a smoother heuristic cooperative mechanism, extending the classical optimization framework by leveraging weighted collaboration. This heuristic cooperative mechanism dynamically adjusts the influence of each actor based on the model's current preferences, enabling more effective data filtering decisions.

In practice, we choose to use the multi-actor collaborative mechanism for data selection. We have added comparisons with single-actor and competitive mechanisms in Table 5 to further elaborate the effectiveness of collaboration.

A.5 Detailed Analysis of Computational Overhead

In this subsection, we compare the computational overhead of our multi-actor collaboration framework with baseline approaches. The analysis focuses on three aspects: **offline computation efficiency**, **online computation efficiency**, and **overall FLOPs requirements**. Table 6 summarizes these comparisons.

A.5.1 Offline Computation Efficiency

Our method achieves superior offline efficiency by requiring only 9.91×10^{19} FLOPs for a one-time

Table 5: This ablation study examines the performance of various combinations of actor collaboration (**Actor**) and dynamic collaborative weight update (**Dynamic**). Accuracy is reported for all tasks, with the highest value in each column shown in **bold**.

		Problem Solving				Commonsense Reasoning				Reading Compreh.		
actor	**Dynamic**	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
with	with	**65.8**	**31.5**	**23.0**	**26.6**	**24.6**	**39.9**	**54.1**	**20.1**	**60.4**	**30.5**	**37.7**
with	without	59.4	26.3	21.3	25.1	20.5	38.9	52.9	19.8	58.1	28.3	35.1
without	-	59.2	26.1	20.3	25.4	21.3	39.1	52.6	**20.1**	56.5	29.1	35.0

Table 6: Comparison of Computational Overhead Across Methods

Selection Method	**Offline Computation Cost (FLOPs)**	**Online Computation Cost (FLOPs)**	**Overall (FLOPs)**
Qu-Rating (Wettig et al., 2024)	7.13×10^{20}	N.A.	7.13×10^{20}
MATES (Yu et al., 2024)	N.A.	1.99×10^{20}	1.99×10^{20}
Multi-actor collaboration (ours)	9.91×10^{19}	1.19×10^{18}	1.00×10^{20}

dataset labeling process using a 109M BERT-based model for inference. This is nearly an order of magnitude more efficient than Qu-Rating, which consumes 7.13×10^{20} FLOPs due to its reliance on a larger 1.3B Sheared-LLaMA model. MATES does not utilize offline computation, relying solely on online updates, which avoids this cost but limits its flexibility and scalability. The offline labeling in our method ensures robust initial scores for large-scale datasets while laying the groundwork for efficient online updates.

A.5.2 Online Computation Efficiency

For adaptive online updates, both our approach and MATES compute influence scores with 1.19×10^{18} FLOPs. However, MATES involves labeling the entire dataset with a 109M BERT-based model in every round, amounting to 1.98×10^{20} FLOPs across four data selection stages. In contrast, our method avoids re-labeling the entire dataset, significantly reducing the computational cost by focusing on labeling the large pretraining datasets only once.

Overall, our approach cuts the computational cost in half compared to MATES and requires only about 1/7 of the computational resources used by Qu-Rating.

A.6 Analysis of Ablation Study

A.6.1 Analysis of actor Roles on Different Type of Tasks

We show the actor ablation study conducted on 373M LLaMA2 models Table 7 as well as 1.3B LLaMA2 models Table 2.

First, the ablation study underscores *the importance of each actor in achieving optimal performance* across the training tasks. When the quality, domain, and topic actors are used together, the model performs best, highlighting the benefits of their combined use, as shown in Table 2. In terms of evaluating *the performance of each actor*, we find that the quality actor outperforms other single-actor configurations, excelling in problem solving tasks. However, its performance drops on tasks requiring domain knowledge or contextual understanding. Here, the domain and topic actors play a crucial role, as they excel in these areas. Despite this, neither performs well on problem solving tasks, except for the topic actor, which significantly improves MMLU performance, indicating that topic diversity may benefit such tasks. In terms of evaluating *the combination of the actors*, we find that removing any actor noticeably reduces overall accuracy, though the impact varies. Excluding the quality actor leads to the largest drop, significantly affecting performance in problem solving tasks, and commonsense reasoning tasks like OpenbookQA. This highlights the quality actor's vital

Table 7: This ablation study examines the performance of various combinations of actor collaboration and update mechanisms. All models are in 373M LLaMA2 architecture. Accuracy is reported for all tasks, with the highest value in each column shown in **bold**.

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Quality&Domain&Topic actor	**57.9**	**24.7**	**21.9**	25.4	**20.2**	**37.9**	**52.6**	**20.4**	59.6	**29.4**	**35.0**
without collaboration update	47.9	20.4	21.0	25.1	17.2	37.3	51.3	20.0	56.5	28.3	32.5
Domain&Quality actor	55.1	18.6	21.7	24.4	17.4	37.1	51.2	19.8	**61.7**	28.2	33.5
Topic&Quality actor	56.2	24.4	21.8	25.2	19.4	36.3	49.0	19.7	56.1	28.5	33.6
Domain&Topic actor	44.6	18.3	21.7	25.7	16.2	36.6	51.9	19.9	61.6	27.8	32.4
Quality actor	53.0	24.7	21.8	25.5	18.0	36.3	49.5	18.1	57.0	28.0	32.9
Domain actor	44.1	19.1	20.8	25.6	16.6	36.8	52.0	19.7	56.7	28.2	32.0
Topic actor	42.7	19.2	21.0	**27.0**	17.4	37.1	50.7	19.7	54.6	28.5	31.8
No actor	42.5	20.0	21.1	23.8	14.6	35.9	50.1	18.8	56.1	27.9	31.1

role in reasoning and problem-solving. Similarly, excluding the topic actor causes a performance drop in ARC-Challenge and a significant reduction in MMLU, emphasizing its importance in tasks covering diverse subjects; removing the domain actor results in a performance drop in commonsense reasoning tasks, underscoring its key contribution to these areas. Second, the ablation study verifies the design and implementation of *the collaborative dynamic adjustment of the actors' weights* (introduced §3.4) for efficient data selection. Concretely, in this variant, all actors were initialized with equal weights, which remained fixed throughout training without adjusting for individual actor performance. Surprisingly, this fixed-weight approach (equal to random sampling) resulted in a significant drop in overall performance compared to the dynamic weighting used in the collaborative update framework, as shown in Table 2. We believe this result from the ablation study is a strong indicator that the dynamic adjustment of the celebration mechanism is essential for efficient data selection.

A.6.2 Ablation Study on Reference Tasks Selection

In our experiments of selecting reference tasks in Table 8, we observe that while the choice of reference tasks can influence performance, the impact on average performance is marginal (within 0.5 points). Using different reference tasks consistently leads to a significant improvement in average performance compared to random data selection, demonstrating that our method is not sensitive to the choice of reference tasks.

A.7 Details of Pretraining

A.7.1 Details of Pretraining Models Architecture

The specific architecture of pretraining model is shown in Table 9. Each model was trained on 32x NVIDIA A800, employing a global batch size of 4×2^{20} tokens and completing 7,500 steps in about 14 hours. The average token processing rate per GPU was about 20,000 tokens per second. The learning rate was set to 5×10^{-5}, and the Adam optimizer was employed with hyperparameters ($\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 10^{-8}$).

A.7.2 Details of Baseline Method Implementation

Regarding the classifier methods, QuRating (Wettig et al., 2024) and DSIR (Xie et al., 2023b), we implement QuRating by downloading the open-source checkpoint from Hugging Face. Notably, the released model has a context length of 4096, whereas ours is 1024. However, this discrepancy does not impact our testing tasks, as our maximum of 5-shot examples remains within the 1024 limit. Despite this, we have totally similar model configuration as well as the total number of training tokens with all the checkpoints we downloaded. Similarly, the replication of PPL is based on the publicly available checkpoint from the original paper. For FineWeb-Edu (Lozhkov et al., 2024), we download the quality scorer to label all the training data from SlimPajama datasets, and adopt the methodology described in the corresponding publication and train all the model from scratch.

Domain mixing refers to the technique of combining data from different sources or domains to

Table 8: This ablation study examines the performance of various combinations of reference task. Accuracy is reported for all tasks, with the highest value in each column shown in **bold**.

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Reference Tasks	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
LAMBADA&SQuAD&Jeopardy	**65.8**	**31.5**	23.0	26.6	24.6	39.9	54.1	20.1	**60.4**	**30.5**	**37.7**
LAMBADA	64.3	31.2	22.3	**26.8**	23.5	39.6	**54.6**	20.4	59.6	30.1	37.2
SQuAD	65.1	30.9	**23.4**	25.9	**24.9**	40.1	53.8	21.2	59.1	29.3	37.4
Jeopardy	63.9	30.3	23.6	26.3	24.1	**40.7**	54.5	**21.8**	59.1	30.2	37.5
Random selection	54.6	23.0	22.1	24.9	18.8	40.3	52.9	21.5	53.0	29.8	34.1

Table 9: Architecture of pre-trained decoder-only models.

Hyperparameter	370M Model Value	1.3B Model Value	3.6B Model Value	8B Model
Vocabulary Size	32,000	32,000	128,256	128,256
MLP Ratio	8/3	8/3	8/3	3.5
Hidden Dimension Size	2048	1024	3072	4096
Number of Layers	24	24	28	32
Number of Attention Heads	16	8	24	32
Number of KV Attention Heads	16	8	8	8
RoPE Base	10,000	10,000	500,000	500,000
Maximum Context Window Length	1024	1024	1024	1024
Number of Parameters	373,867,520 (370M)	1,345,423,360 (1.3B)	3,606,752,256 (3.6B)	8,030,261,248 (8B)

enhance the diversity and robustness of a model's training dataset. In our implementation, we apply various mixing methods: DoReMi (Xie et al., 2023a), DOGE (Fan et al., 2024), DMLaw (Ye et al., 2024), and RegMix (Liu et al., 2024). Each method contributes distinct proportions of data from specific domains, as reflected in the domain weights presented in Table 10. Notably, the weights indicate the percentage of contributions from each domain.

For the reproduction of MATES (Yu et al., 2024), we start by utilizing Random-Slimpajama at the 1500th training step as our primary pretraining model and fine-tune the BERT-base from the original thesis as our data influence model. During the training of the data influence model, we uniformly sample 1/13 of the data as hold-out data from each area of our dataset and employ LAMBADA (Paperno et al., 2016) as a reference task, following the MATES methodology. Ultimately, we use the trained BERT-base data influence model to predict the entire training dataset, selecting the top 1/20 as our pretraining data. This selection process is executed using the Gumbel-Top-k algorithm (Kim et al., 2016), consistent with MATES. We leverage a four-step updates similar to the original paper, and conduct the above implementation at 1500th, 3000th, 4500th and 6000th model training steps using the current models.

A.7.3 Details actor Weight Initialization

For the *domain actor*, we use the document's meta-information, label the data with domain information and save this into the domain actor's memory, where the domain $\text{Domain}(x_i)$ belongs to one of ArXiv, Book, Wikipedia, CommonCrawl, GitHub, StackExchange, C4.

For *quality actor*, we adopt the FineWeb-Edu quality scorer (Lozhkov et al., 2024), which is fine-tuned as a BERT-like regression model (Merrick et al., 2024) using Llama3-70B-Instruct annotated 500k examples. This will give out a successive quality score $\text{Quality}(x_i) \in \mathcal{R}^{[0,5]}$ with higher score represent higher quality. We then map this score into five quality intervals $\{I_j\}_{j=1}^{5}$, as

$$\text{Quality}(x_i) \mapsto I_j = \begin{cases} [j-1, j), & \text{if Quality}(x_i) \in [j-1, j), \\ & j = 1, 2, 3, 4 \\ [4, 5], & \text{if Quality}(x_i) \in [4, 5], \\ & j = 5 \end{cases}$$

We store the quality interval corresponding to each data point in the quality actor's memory.

For the *topic actor*, due to the absence of a suitable pretrained model for topic classification and labeling, we designed a classification schema using 1.44 billion documents collected by the Common Crawl project (Project, 2007) and fine-tuned a BERT-like regression model on 500k GPT-4o annotated samples, the overall pipeline is depicted in

Figure 6: We illustrate the distribution of manually annotated and clustered data, which includes 13 topics: Infrastructure, Law and Government, Networking, Activity, Business and Industry, Nature, Literature and Art, Education, Finance, Technology, Entertainment, Health, and Others.

Figure 4. Further details on the topic classification approach and BERT model training are provided in §A.2. Using this topic classifier, we categorize each document into one of 13 topics: Activity, Education, Entertainment, Finance, Health, Business and Industrial, Infrastructure, Literature and Art, Nature, Others, Law and Government, Networking, Technology, and store the topic information in the topic actor's memory.

We employ an actor weight initialization tech-

Table 10: Exact domain weights (%) on SlimPajama obtained by data mixing methods. Abbreviations: C.C. = CommonCrawl, Wiki = Wikipedia, StackEx. = StackExchange

Mixing Method	C.C.	C4	GitHub	Books	ArXiv	Wiki	StackEx.
SlimPajama	52.20	26.70	5.20	4.20	4.60	3.80	3.30
DoReMi	38.11	11.41	6.54	8.19	4.24	23.07	8.47
DOGE	21.35	26.93	7.03	4.50	8.80	14.82	16.58
DMLaw	12.50	25.00	14.06	9.38	25.00	1.56	12.50
RegMix	17.37	51.03	0.23	0.23	0.08	29.77	1.27

nique within the RegMix (Liu et al., 2024) framework, which is crucial for the effective training of proxy models. Our dataset is organized into three distinct categories: domain, quality, and topic. For each category, we initialize the data weights based on the original proportions across 512 configurations and subsequently train a TinyLlama-1M with 1 billion tokens as a proxy model for each configuration. We evaluate this model on previously unseen data mixtures, specifically using validation set loss, following RegMix, for assessment. We then fit a regression model based on the performance results of the 512 proxy models to predict the optimal data mixture for training large-scale LMs. The results of the LightGBM regression analysis and Spearman correlation of the loss prediction performance are presented in 7.

Upon training the regression model, we systematically investigate the entire spectrum of potential data mixtures by utilizing the trained model to predict the target values for each candidate mixture. This process allows us to identify the input that produces the optimal target value. Following the simulation and identification of the most effective data mixture, we then generalize this top-ranked configuration for large-scale model training, incorporating a significantly larger volume of tokens.

A.8 Full Experimental Results

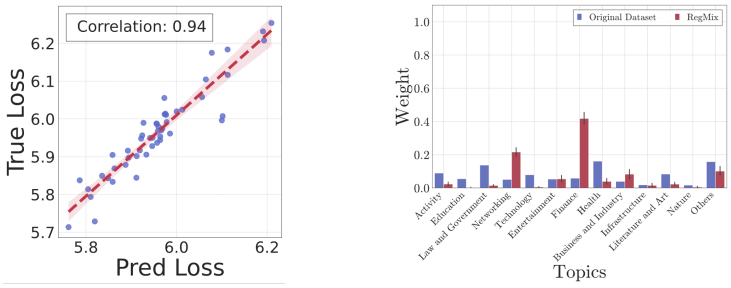
We show the full results of all tasks in Table 11, Table 12 and Table 13. In analyzing the full experiment results, it is evident that our model consistently outperforms other methods across various tasks. Overall, for the zero-shot scenario, the classifier method outperforms the influence function in terms of average performance, while domain mixing yields the poorest results. Our method achieves an impressive average accuracy of 36.5, significantly surpassing the next best classifier, QuRating's series, which scores 35.5. This underscores the robustness of our approach, particularly in challenging problem-solving domains such as ARC-C, ARC-E, and MMLU, where we exceed competing models by considerable margins.

Our model demonstrates superior performance in the three-shot scenario, achieving an impressive average accuracy of 37.7, thereby maintaining its lead. Notably, we excel in the ARC-E and ARC-C benchmarks, attaining scores of 65.8 and 31.5, respectively, which highlights our model's effective utilization of few-shot learning. In contrast, the leading alternative methods underperform, particularly in more complex tasks such as MMLU and BoolQ.

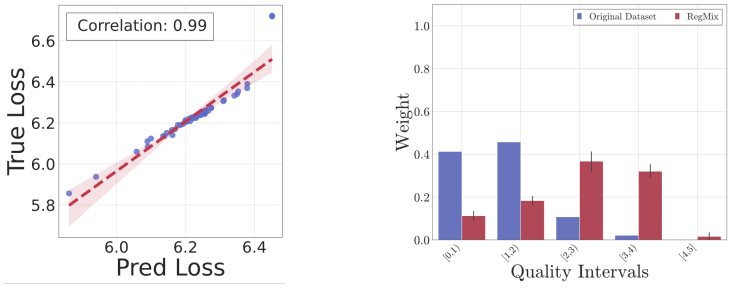
In the five-shots evaluation, our model continues to demonstrate competitive performance, with scores reflecting a consistent trend of superiority across various domains, while other non-leading methods also maintain high levels. These results underscore our model's robust capacity to generalize across diverse question-answering tasks, affirming its advantages over conventional classifiers and highlighting its potential for practical applications in real-world scenarios.

A.9 Implementation Details of our Methods

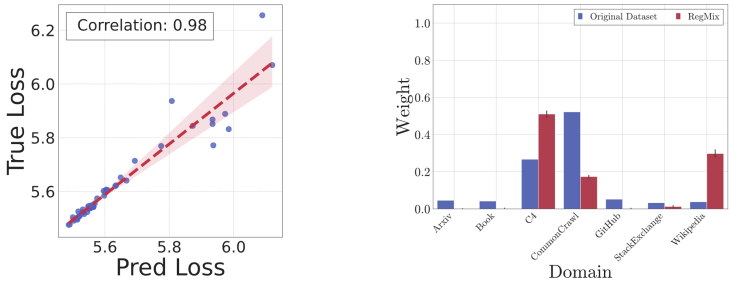
To further refine the model's performance, we calculate rewards for each sampled data point by approximating the rewards using influence functions, as shown in Equation 3. Following (Engstrom et al., 2024), we choose LAMBADA (Paperno et al., 2016), SQuAD (Rajpurkar, 2016) and Jeopardy (Tunguz, 2019) as reference tasks. We followed methods provided in (Engstrom et al., 2024), (Xia et al.) and (Park et al., 2023) to calculate the Hessian and the gradients in the influence functions. In our implementation, we project gradients into an 8,192-dimensional space for both the validation and training datasets. To optimize the gradient computation process, we divide each data



(a) Topic



(b) Quality Interval



(c) Domain

Figure 7: We present the results of the LightGBM regression analysis and Spearman correlation regarding the loss prediction performance and the weights of each candidate data-(a) Topic, (b) Quality Interval and (c) Domain after mixture across all categories.

category into eight slices, thereby enabling parallel computation across eight GPUs. Each slice contains 1,250 data points. After calculating gradients for each slice, the results are concatenated in their original sequence to ensure data integrity. This slicing strategy not only accelerates the processing by utilizing GPU parallelism but also maintains consistency in gradient calculation. Additionally, for the validation datasets, we uniformly sample 500 data points to ensure a balanced evaluation procedure. All prompts across the datasets are carefully aligned to maintain task coherence, a crucial factor in multi-task learning scenarios. Furthermore, we implement a sliding window of 1,024 tokens with a 256-token overlap to ensure consistent tokenization across the entire dataset. This sliding window technique efficiently extracts a maximum of 1,024 tokens from each data point, ensuring uniform encoding across different datasets and tasks, thus improving the overall consistency and reliability of the data processing pipeline.

A.10 Details of Influence Changes During Different Pretraining Stages

We present the details of influence change during the pretraining process for domain (Figure 8), quality intervals (Figure 9) and topic (Figure 10).

A.11 Data Distribution Analysis of the SlimPajama Dataset

We finally present the data distribution analysis of the SlimPajama dataset from three dimensions: topic, domain and quality intervals, as Figure 11 to Figure 13 shows.

Table 11: Table Showing Various Selection Methods and Their Scores with Changes. We report accuracy for all tasks, and **bold** the best result in each column. Abbreviations: O.B.QA = OpenbookQA W.G. = WinoGrande, C.S.QA = CommonSenseQA, Compreh. = Comprehensions

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Random sample											
Uniform-30B	54.3	23.4	22.3	23.9	18.6	39.8	52.8	19.2	55.4	30.0	34.0
Uniform-60B	55.2	24.6	22.5	23.4	21.0	39.7	51.9	19.5	59.8	**33.1**	35.1
Perplexity-based data selection											
PPL	49.3	20.1	22.4	23.6	16.2	36.0	48.1	18.8	61.4	29.3	32.5
Classifier-based data selection											
QuRating-Facts	56.1	23.3	22.4	24.8	21.6	39.2	54.1	19.9	61.5	31.6	35.5
QuRating-Req	54.9	24.4	23.2	25.2	21.4	38.1	54.5	20.6	**61.6**	31.3	35.5
QuRating-Writing	53.6	23.2	**23.4**	23.2	21.0	38.1	52.8	19.7	59.4	31.6	34.6
QuRating-Edu	57.0	24.4	22.0	25.0	20.4	**40.3**	53.7	20.2	60.1	32.2	35.5
FineWeb-Edu	53.8	23.4	21.8	23.9	19.8	39.2	51.7	20.8	59.7	32.0	34.6
DSIR-Book	45.4	20.8	22.0	23.0	18.8	39.9	**54.6**	19.7	58.3	30.8	33.3
DSIR-Wiki	50.6	21.1	21.6	23.0	19.2	36.6	53.0	19.8	60.5	29.2	33.5
Domain mixing methods											
DOGE	49.4	21.8	22.5	23.0	18.0	38.0	52.7	19.9	60.0	30.0	33.5
DoReMi	50.1	20.2	22.5	23.7	17.8	38.7	52.8	19.7	58.6	30.8	33.5
DMLaw	49.6	21.9	23.2	23.6	17.8	38.6	51.8	20.1	60.4	29.0	33.6
RegMix	50.0	22.3	22.1	22.9	18.8	38.0	52.8	19.9	58.9	31.2	33.7
Influence functions											
MATES	50.0	21.4	22.7	25.3	19.0	39.8	53.6	**21.3**	59.9	32.1	34.5
Multi-actor Collaboration (Ours)	**61.1**	**28.2**	22.6	**26.0**	**24.4**	38.2	54.2	19.5	61.0	29.8	**36.5**

Table 12: Table showing various selection methods and their three-shots performance. We report accuracy for all tasks, and **bold** the best result in each column. Abbreviations: O.B.QA = OpenbookQA W.G. = WinoGrande, C.S.QA = CommonSenseQA, Compreh. = Comprehensions

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Random sample											
Uniform-30B	54.6	23.0	22.1	24.9	18.8	40.3	52.9	**21.5**	53.0	29.8	34.1
Uniform-60B	58.8	25.5	23.0	**27.2**	20.0	**41.8**	53.6	19.6	56.9	**32.7**	35.9
Perplexity-based data selection											
PPL	50.6	21.3	22.7	25.2	15.6	37.7	48.9	20.1	61.5	22.3	32.6
Classifier-based data selection											
QuRating-Facts	59.5	25.7	22.6	25.9	19.8	40.2	**54.6**	19.2	60.8	24.8	35.3
QuRating-Req	59.3	25.9	22.7	26.1	19.6	39.7	53.7	20.5	58.5	22.7	34.9
QuRating-Writing	56.9	25.7	**23.1**	26.0	20.4	41.1	53.6	20.2	51.4	22.6	34.1
QuRating-Edu	60.8	26.5	22.5	26.7	20.2	41.4	54.6	20.6	55.5	22.7	35.1
FineWeb-Edu	56.2	25.7	22.3	26.2	20.6	40.1	50.5	19.7	56.6	31.4	34.9
DSIR-Book	48.7	21.0	22.6	25.6	18.6	42.5	53.7	19.5	57.9	22.9	33.3
DSIR-Wiki	53.2	22.4	22.6	25.3	17.6	37.1	52.7	21.4	**61.6**	24.2	33.8
Domain mixing methods											
DOGE	52.4	21.9	22.4	27.0	17.4	39.9	52.0	18.2	57.8	29.8	33.9
DoReMi	53.2	21.4	22.2	24.7	18.2	38.4	50.9	20.6	59.7	31.1	34.0
DMLaw	51.5	21.4	22.4	25.2	18.2	39.0	50.7	19.4	52.6	29.8	33.0
RegMix	53.1	22.1	22.2	25.4	19.0	39.1	53.5	18.4	60.7	30.0	34.4
Influence functions											
MATES	52.6	21.8	22.6	26.7	20.4	40.9	53.7	19.7	57.6	31.8	34.8
Multi-actor Collaboration (Ours)	**65.8**	**31.5**	23.0	26.6	**24.6**	39.9	54.1	20.1	60.4	30.5	**37.7**

Table 13: Table showing various selection methods and their five-shots performance. We report accuracy for all tasks, and **bold** the best result in each column. Abbreviations: O.B.QA = OpenbookQA W.G. = WinoGrande, C.S.QA = CommonSenseQA, Compreh. = Comprehensions

	Problem Solving				Commonsense Reasoning				Reading Compreh.		
Selection Method	**ARC-E**	**ARC-C**	**MathQA**	**MMLU**	**O.B.QA**	**SIQA**	**W.G.**	**C.S.QA**	**BoolQ**	**RACE**	**Average**
Random sample											
Uniform-30B	54.5	21.9	22.4	25.6	19.2	39.7	54.2	19.7	53.2	30.8	34.1
Uniform-60B	59.1	26.0	22.4	**26.9**	21.6	**42.1**	54.3	21.0	55.7	**32.4**	36.2
Perplexity-based data selection											
PPL	49.2	21.2	22.5	24.9	14.6	36.7	49.8	20.6	60.6	23.3	32.4
Classifier-based data selection											
QuRating-Facts	60.5	25.4	**23.4**	26.4	20.2	40.3	51.9	19.0	58.0	23.3	34.8
QuRating-Req	59.9	26.4	22.8	25.6	21.8	40.1	53.7	19.6	56.9	22.3	34.9
QuRating-Writing	57.3	25.3	22.6	25.0	21.4	41.6	53.5	19.6	49.5	22.1	33.8
QuRating-Edu	60.8	26.5	22.5	26.5	20.2	41.4	**54.6**	**21.1**	55.5	22.7	35.2
FineWeb-Edu	56.6	24.9	22.6	25.8	19.8	39.4	51.2	19.7	55.9	30.9	34.7
DSIR-Book	49.7	21.1	22.1	25.6	19.8	41.7	54.1	18.3	55.6	22.9	33.1
DSIR-Wiki	53.6	22.3	23.0	25.3	17.6	36.7	52.2	20.4	60.2	22.6	33.4
Domain mixing methods											
DOGE	53.0	21.8	22.0	26.3	17.2	40.1	51.7	18.8	58.5	30.1	33.9
DoReMi	52.7	22.2	22.4	25.5	16.2	39.3	51.9	20.9	60.0	31.0	34.2
DMLaw	52.4	21.4	23.0	25.7	17.2	39.2	50.6	19.2	51.4	29.9	33.0
RegMix	53.5	24.0	21.2	25.0	19.6	41.0	53.2	19.0	**61.3**	30.2	34.8
Influence functions											
MATES	53.6	21.6	22.6	26.1	20.4	41.7	53.1	20.4	60.1	32.0	35.2
Multi-actor Collaboration (Ours)	**64.9**	**31.1**	22.4	26.3	**23.6**	39.0	53.1	20.4	60.4	30.7	**37.2**

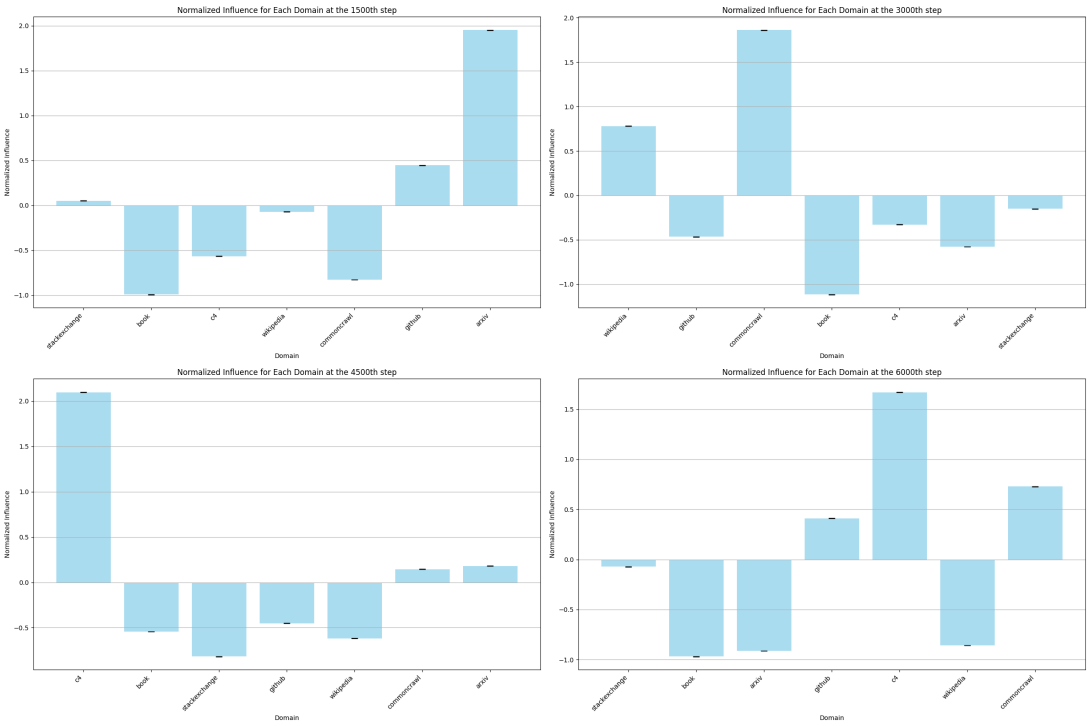


Figure 8: We present the normalized influence for each domain across various training steps.

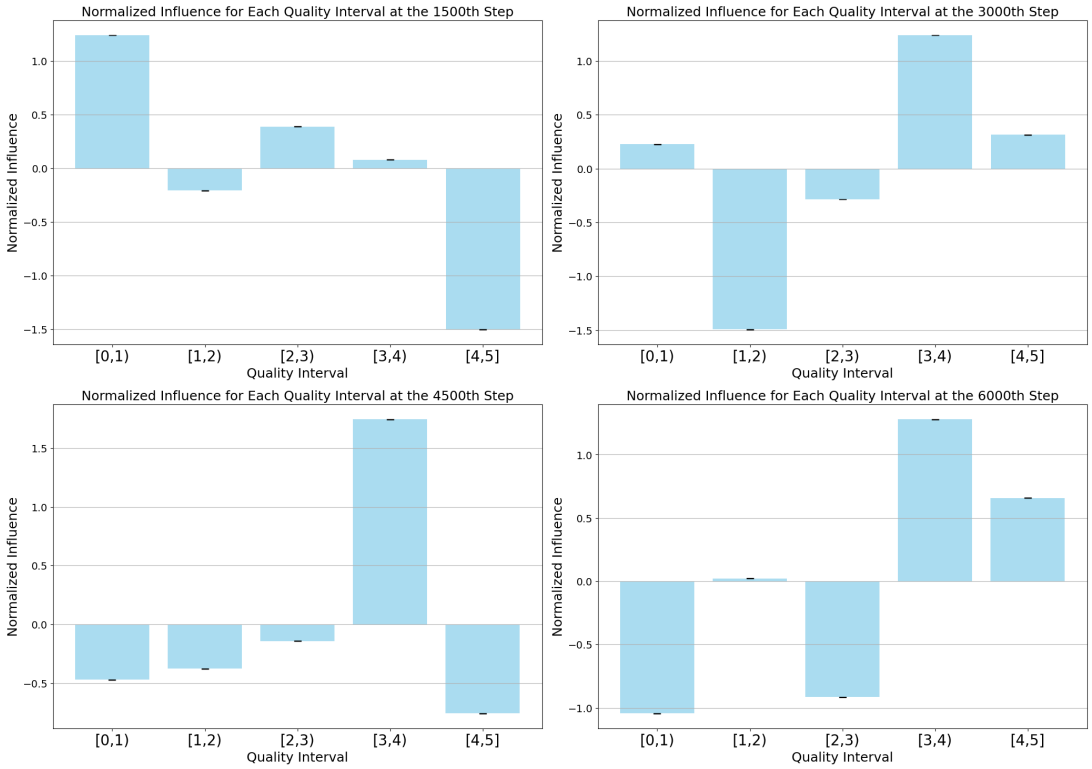


Figure 9: We present the normalized influence for each quality interval across various training steps.

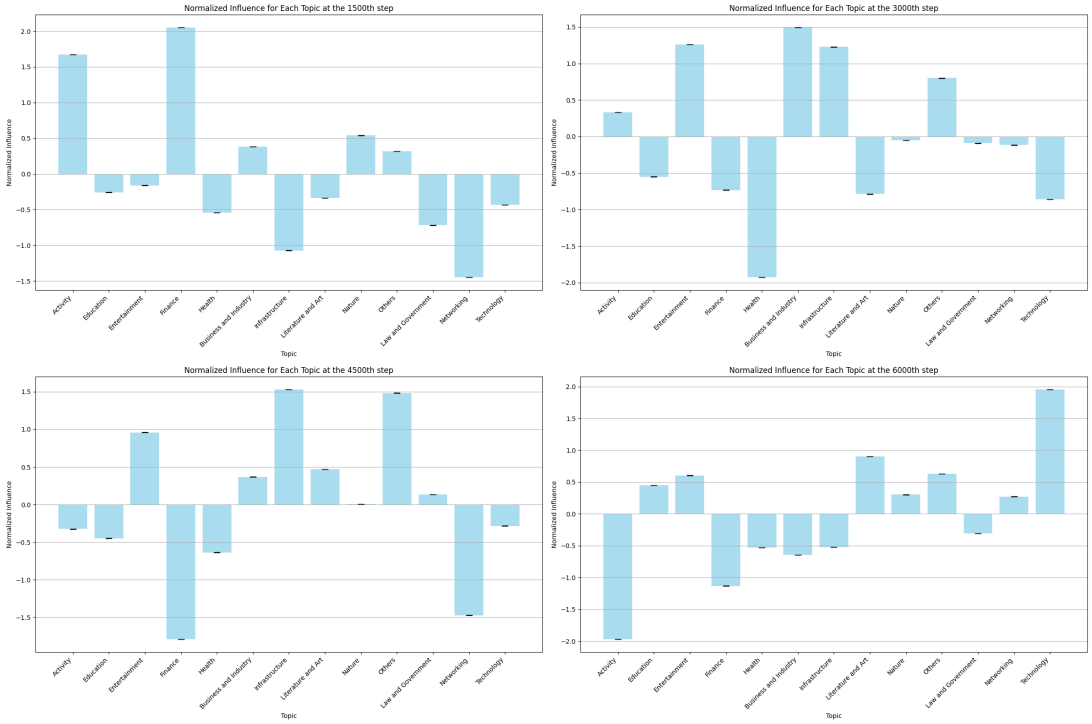


Figure 10: We present the normalized influence for each topic across various training steps.

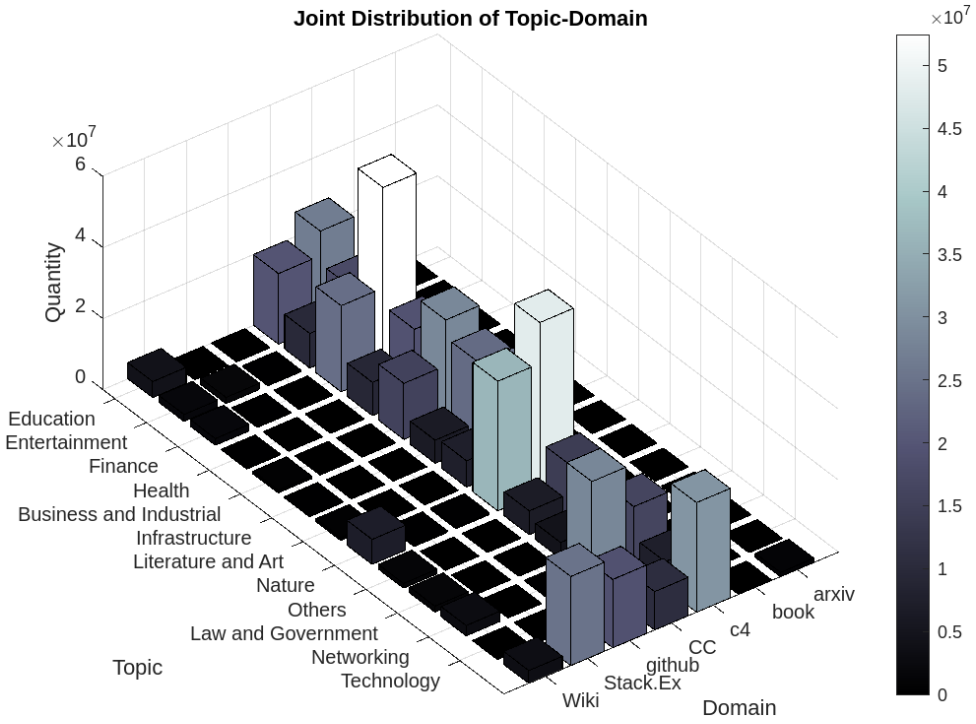


Figure 11: The illustration of the joint distribution of topics and domains.

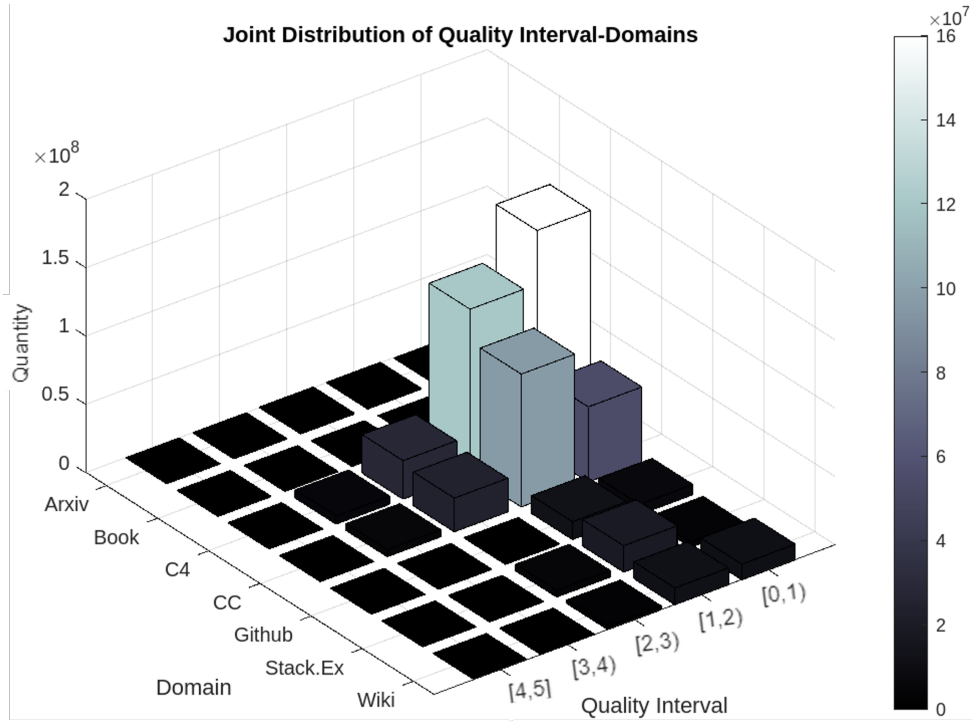


Figure 12: The illustration of the joint distribution of quality intervals and domains.

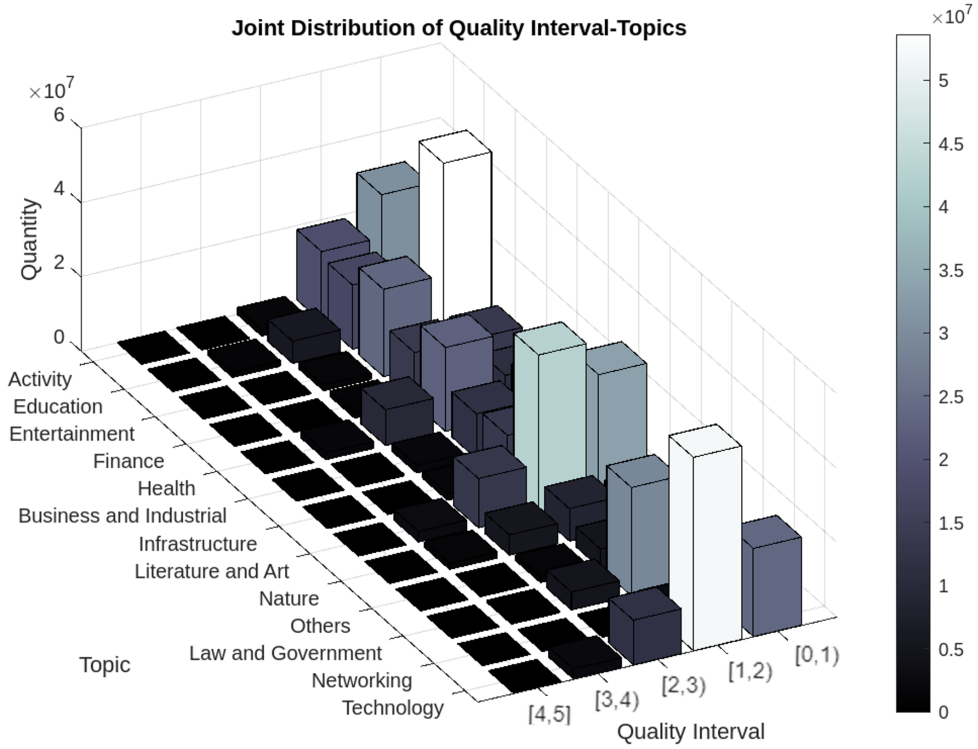


Figure 13: The illustration of the joint distribution of quality intervals and topics.