Beyond Ranking: Role-Specific Signals for Token-Budgeted SFT Data Selection

Haocheng Zhu[1] **Tianqi Wang**[1] **Jun Suzuki**[1,2,3]
[1]Tohoku University [2]RIKEN [3]NII LLMC
is-failab-research@grp.tohoku.ac.jp

Abstract

Supervised fine-tuning (SFT) is a core stage in post-training general-purpose language models, but modern SFT training corpora are often large, heterogeneous mixtures, making data selection important under fixed token budgets. Simple ranking approaches can be informative yet insufficient: ranking alone does not determine how to allocate a fixed token budget across examples. We study the role-specific use of likelihood-derived signals for token-budgeted subset construction. An external instruction-tuned reference model computes response-side negative log-likelihood scores to rank training examples. We then combine this ranking with base model likelihood filtering to restrict the candidate region and response-side cosine filtering to reduce redundancy in the selected subset. In the OLMo fine-tuning setting studied here, ranking examples only by reference-model likelihood underperforms the random baseline, whereas the same ranking becomes more effective when combined with base model likelihood filtering and cosine-based redundancy control. Our results support viewing token-budgeted SFT data selection as a budget-allocation problem rather than only as an example-ranking problem.

1 Introduction

Post-training has become a central stage in building general-purpose language models. Recent model releases such as Llama 3 and Qwen 3 commonly provide instruct models alongside base models. Open post-training efforts such as Tulu 3 further make this process more reproducible by releasing training data, code, recipes, and evaluation tools. Together, these trends highlight the practical importance of post-training in modern language-model development (Ouyang et al., 2022; Llama Team, 2024; Yang et al., 2025; Lambert et al., 2025). Among these stages, supervised fine-tuning (SFT) teaches a pretrained model to follow instructions and often serves as the starting point for later alignment stages. Prior work has also shown that SFT effectiveness depends strongly on the quality, style, and construction of the training data (Zhou et al., 2023a; Zhao et al., 2026). Therefore, as SFT datasets have grown from small curated collections to large heterogeneous mixtures, training on all available examples becomes increasingly expensive and is not always desirable.

This motivates the following SFT data selection problem: given a large candidate pool, how can we construct a compact subset that preserves the utility of full-data SFT under a fixed token budget?

For SFT, negative log-likelihood (NLL)-based scores are appealing because they are easy to compute, require no additional annotation, and are directly tied to the supervised training objective. A natural approach is therefore to score each example, sort the data pool, and allocate the token budget to the highest-ranked examples. However, recent studies show that such score-based selection does not consistently outperform randomly selected subsets across data pools, budgets, and evaluation settings (Diddee and Ippolito, 2025; Ivison et al., 2025; Xia et al., 2025). A plausible explanation is that a single ranking score does not control the composition of the final subset. Random selection can preserve diversity in the original mixture data pool, while top-ranked examples can be redundant or biased toward particular formats and response styles. Thus, NLL scores may be informative, but using them only as standalone ranking scores is insufficient for token-budgeted subset construction. This motivates our central question: *How should NLL scores be used to construct, from a large instruction mixture data pool, a compact SFT subset that maintains performance on held-out evaluation under a fixed token budget?*

We study this question by assigning different signals to different roles in subset construction. Fig-

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 41717–41729
October 24-29, 2026 ©2026 Association for Computational Linguistics

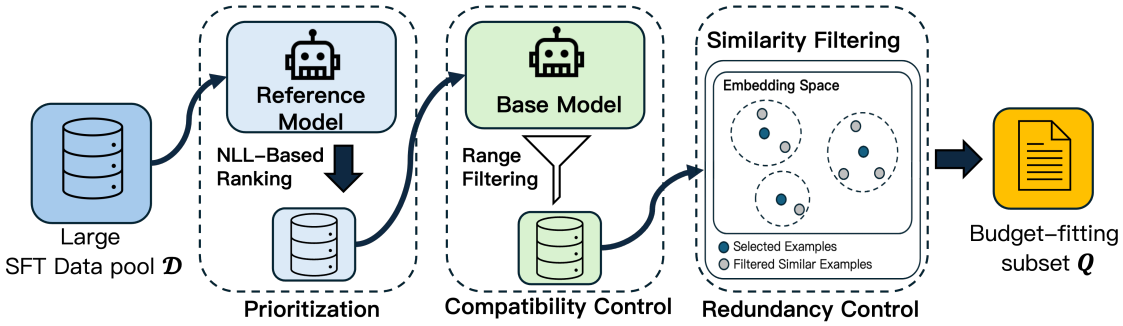


Figure 1: Overview of our role-separated framework.

ure 1 illustrates our role-separated framework. The base model is the pretrained model that we ultimately fine-tune, and the reference model is a separate instruction-tuned model used only for scoring examples. We use these two models for different purposes. In our experiments, these two models come from different model families and have different scales. The external instruction-tuned reference model provides a complementary instruction-following view of the data, while the pretrained base model measures how each example appears to the model that will actually be fine-tuned.

Our method starts by using the reference model to compute response-side NLL scores. We refer to these scores as **RefNLL** scores and use them to rank the examples. This ranking gives the greedy selection procedure a preference over which examples should be considered earlier. Next, the base model computes NLL scores for all the examples, which we refer to as **BaseNLL**. We use these scores to restrict the eligible candidate region. A very low BaseNLL score indicates that the response is already easy to predict, whereas very high BaseNLL indicates that the response is highly unexpected for the base model. Finally, response-side cosine filtering is applied greedily along the ranked list to avoid adding examples that are too similar to examples already selected. In this way, likelihood-derived signals are used for prioritization and compatibility control, while an embedding-based similarity signal is used for redundancy control.

Our experimental results show that RefNLL ranking alone can perform worse than random selection. When combined with the BaseNLL range filter and cosine filtering, RefNLL ranking yields compact subsets that preserve much of the utility of full-data training.

Our contributions are:

- We revisit likelihood-based SFT data selection in a large heterogeneous instruction mixture.
- We propose and analyze a role-separated subset-construction strategy that combines RefNLL ranking, BaseNLL filtering, and response-side cosine filtering.
- We evaluate the full role-separated configuration and perform component ablations to isolate the contributions of compatibility and redundancy control under a fixed token budget.

2 Related Work

2.1 Supervised Fine-Tuning

Supervised fine-tuning (SFT) adapts a pretrained language model to follow human instructions by training on paired examples, and has become a standard stage in modern LLM training pipelines (Wei et al., 2022; Ouyang et al., 2022). Recent model releases, including Llama 3 and Qwen 3, increasingly include supervised fine-tuned or instruction-tuned variants alongside the base models. At the same time, open post-training efforts such as Tulu 3 organize SFT as part of a broader pipeline connecting data construction, fine-tuning, alignment, and evaluation (Llama Team, 2024; Yang et al., 2025; Lambert et al., 2025). In such practical post-training pipelines, the SFT data pool is often large and heterogeneous. Training on all available examples can therefore be costly and may not be the most efficient use of the SFT budget. This motivates SFT data selection: selecting a compact subset that preserves most of the utility of full-data SFT under a small budget (Xia et al., 2025; Zhao et al., 2026).

2.2 SFT Data Selection

Classical data selection often scores examples by how well they match a target distribution, with

cross-entropy-difference methods being a representative approach in domain adaptation (Moore and Lewis, 2010; Axelrod et al., 2011). A complementary concern is redundancy: high-scoring examples may provide overlapping information, so constructing a representative subset also needs to consider coverage and non-redundancy (Lee et al., 2022; Suzuki et al., 2023).

Recent works on SFT data selection for large language models extend these ideas in several directions. Small-data studies such as LIMA and LIMO emphasize the effectiveness of carefully curated examples (Zhou et al., 2023a; Ye et al., 2025). Automatic methods such as LESS, SelectIT, ROSE, and D3 explore various data selection paradigms including influence-based, uncertainty-based, reward-oriented, and multi-criterion (Xia et al., 2024; Liu et al., 2024; Wu et al., 2025; Zhang et al., 2025a). However, large-scale studies show that such strategies do not consistently outperform random selection (Xia et al., 2025; Diddee and Ippolito, 2025; Ivison et al., 2025). Motivated by prior work, we study how NLL scoring and redundancy control can be combined for SFT subset construction.

3 Method

Let $D = \{z_i\}_{i=1}^{N}$ be an SFT data pool, where each example $z_i = (x_i, y_i)$ consists of an instruction-side context x_i and an assistant response y_i. Given a pretrained base model F_b, our goal is to select a subset $Q \subset D$ under a reduced token budget while maintaining SFT performance for F_b. Q is constructed under a token budget B, which counts all training tokens after applying the chat template, including both the instruction-side context and the assistant response. Therefore, the selected subset must satisfy

$$\sum_{z \in Q} c(z) \leq B,$$

where $c(z)$ denotes the number of tokens in the serialized training example z.

As illustrated in Figure 1, our role-separated framework consists of three components. Examples are ranked by the prioritization component. Next, extreme examples are filtered out by the compatibility control component. We identify prioritized eligible examples jointly through these two stages. Finally, with redundancy control, we remove similar examples to preserve the diversity of the subset.

3.1 Prioritization

To rank the examples, the prioritization component produces ranking signals based on negative log-likelihood (NLL) scores. For a causal language model M, we define the supervised-token average NLL of z_i as

$$\ell_M(z_i) = \frac{1}{|Y_i^{(M)}|} \sum_{t \in Y_i^{(M)}} -\log p_M(w_{i,t}^{(M)} \mid w_{i,<t}^{(M)}),$$

where $w_i^{(M)}$ is the full token sequence obtained by applying the chat template associated with model M and tokenizing with the tokenizer of the model M, and $Y_i^{(M)}$ denotes the assistant-response token positions that contribute to the SFT loss. Equivalently, these are the token positions over which we compute the supervised-token NLL. Note that the instruction-side tokens are used as conditioning context but are not included in the NLL computation. Unless otherwise stated, all NLL scores in this paper are computed in this supervised-token sense.

We refer to F_r as a reference model, and $s_{\mathrm{RefNLL}}(z) = \ell_{F_r}(z)$ is used as the main ranking signal for example z. Given such a ranking signal, examples are sorted in ascending order of the ranking signal and traversed sequentially under the token budget.

Examples assigned lower NLL by the reference model appear earlier in the ranked list. The intuition is that lower NLL under a reference model suggests that the response is more predictable for a model already adapted to instruction-following behavior, and may therefore provide a more stable learning signal for the base model during SFT.

To describe both ends of the ranked list induced by each score, we use *top* and *tail* to denote its beginning and end, respectively. In the following subsections, we focus primarily on the top ranking and study how to use it to construct a more effective token-budgeted subset.

3.2 Compatibility Control

We instantiate compatibility control as a BaseNLL range filter. While the reference model determines which examples are prioritized, we use the base model to determine example compatibility for SFT. Extremely high-NLL examples may be too difficult, noisy, or mismatched for the base model under a limited token budget, whereas extremely low-NLL examples may already be too easy to provide

Algorithm 1 Greedy response-side cosine filtering

Require: Ranked candidate list L, token budget B, cosine threshold γ, token cost $c(z)$, normalized response-side embedding $e(z)$ for each example z

Ensure: Selected subset Q

$Q \leftarrow \emptyset$
$T \leftarrow 0$
for each example $z \in L$ **do**
  if $T + c(z) > B$ **then**
    continue
  end if
  if $Q = \emptyset$ **then**
    $Q \leftarrow Q \cup \{z\}$
    $T \leftarrow T + c(z)$
    continue
  end if
  $m \leftarrow \max_{z' \in Q} \cos(\boldsymbol{e}(z), \boldsymbol{e}(z'))$
  if $m < \gamma$ **then**
    $Q \leftarrow Q \cup \{z\}$
    $T \leftarrow T + c(z)$
  end if
  if $T \geq B$ **then**
    break
  end if
end for
return Q

substantial marginal learning signal. We do not assume that extreme examples are globally harmful or low-quality; rather, we test whether excluding such extremes improves subset utility when only a limited number of tokens can be selected. The eligibility of examples is model-specific; thus we use ℓ_{F_b} to define the extremity thresholds. Given lower and upper thresholds τ_l and τ_u, we define the eligible set $\mathcal{E}_{\tau_l,\tau_u}$ as

$$\mathcal{E}_{\tau_l,\tau_u} = \{z \in D : \tau_l \leq \ell_{F_b}(z) \leq \tau_u\}.$$

Examples outside the range are removed, and only eligible examples are added to Q when they do not exceed the token budget.

In the two-sided setting, we filter out examples with extremely high or extremely low BaseNLL. To isolate the effect of each tail, we also evaluate one-sided variants by setting $\tau_l = -\infty$ or $\tau_u = \infty$.

3.3 Redundancy Control

We finally introduce the third component of our role-separated framework: the response-side cosine filter. After the NLL-based ranking and range filtering, the selected subset can still contain many semantically or stylistically similar responses, causing the token budget to be spent on redundant examples. To address this, we apply a greedy filter based on cosine similarity between response-side embeddings. We use response-side embeddings mainly because the SFT loss is computed over response tokens, which are a more direct source of the parameter-update signal during fine-tuning. Instruction-side tokens serve as conditioning context but are not included in the loss, so response-side similarity is intended to better reflect redundancy in the supervised training signal.

For each example z_i, we use a normalized response-side embedding $\boldsymbol{e}(z_i)$. Given a cosine-similarity threshold γ, the filter is applied greedily during traversal of the ranked candidate list. Starting from an empty selected subset Q, we first add the highest-ranked candidate to Q. Then we process the remaining examples according to the given rank. A candidate example z_i is added to Q only if its maximum cosine similarity to the examples already selected is below γ:

$$\max_{z_j \in Q} \cos(\boldsymbol{e}(z_i), \boldsymbol{e}(z_j)) < \gamma.$$

The traversal continues until the token budget is reached or the candidate list is exhausted. This filter preserves the original rank while preventing newly accepted examples from being too similar to the current selected subset. The details are explained in Algorithm 1.

We apply this filter either directly after the NLL-based ranking or after the BaseNLL range filter.

4 Experiments

4.1 Experimental Setup

Data and budget. We use the Tulu-3 OLMo-2 SFT mixture as our main SFT data pool (Walsh et al., 2025)[1]. After preprocessing, the pool contains 578.7M serialized training tokens, including 351.6M supervised response tokens. Unless otherwise stated, subsets are constructed under a 10% total-training-token budget, corresponding to approximately 57.9M serialized training tokens. The budget is counted after applying the SFT chat template and includes both instruction-side context and assistant-response tokens; therefore, different selection rules can yield different numbers of examples and supervised response tokens.

[1] `allenai/tulu-3-sft-olmo-2-mixture-0225`.

Models and scoring. We use OLMo-3-1025-7B (Team OLMo et al., 2025)[2] as the base model to be fine-tuned and to compute BaseNLL $\ell_{F_b}(z)$, which serves as the compatibility signal for range filtering. RefNLL $\ell_{F_r}(z)$ is computed with the separate instruction-tuned reference model `Qwen/Qwen3.5-0.8B` (Qwen Team, 2026), which is used only for scoring and ranking examples. All NLL scores are computed over supervised assistant-response tokens, following Section 3. For data preprocessing and BaseNLL computation, we use the OLMo-3-1025-7B tokenizer with the chat template from `allenai/Olmo-3-7B-Instruct-SFT`. For RefNLL computation, we use the reference model's native tokenizer and chat template.

Embeddings. For cosine filtering, we use `Qwen/Qwen3-Embedding-4B` as the embedding model (Zhang et al., 2025b). For each training example, we extract all non-empty assistant messages and encode each assistant turn independently. Each assistant turn is truncated to at most 4096 tokens, and the total assistant-side token budget per example is limited to 32768 tokens. We average the retained turn-level embeddings to obtain one response-side example embedding, and L2-normalize it before computing cosine similarities. These embeddings are used in the greedy filtering procedure in Algorithm 1. The ranked candidate list L can be either the RefNLL top ranking directly or the RefNLL top ranking after applying the BaseNLL range filter.

Training. The base model is fine-tuned on the selected subsets using the Open-Instruct SFT pipeline (Wang et al., 2023).[3] All runs use the same training recipe and differ only in the selected training subset. Training is conducted for two epochs with a learning rate of 2×10^{-5}, a linear learning-rate scheduler, a warmup ratio of 0.03, and a maximum sequence length of 4096 tokens. We use bf16 mixed precision, DeepSpeed ZeRO-3, flash attention, and gradient checkpointing on 4 H100 GPUs. The per-device training batch size is 4, with 8 gradient accumulation steps. The training seed is fixed to 8 for all main experiments.

Evaluation. Evaluation is conducted with OLMES, a documented standard for reproducible language-model evaluation[4] (Gu et al., 2025). We evaluate the performance on eight benchmarks covering instruction following, reasoning, knowledge, reading comprehension, and truthfulness. Specifically, IFEval evaluates instruction-following ability (Zhou et al., 2023b); BBH evaluates challenging reasoning tasks (Suzgun et al., 2023); DROP evaluates reading comprehension with discrete reasoning over paragraphs (Dua et al., 2019); MMLU evaluates broad multitask knowledge and problem-solving ability (Hendrycks et al., 2021a); GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) evaluate mathematical reasoning, with MATH evaluated using the Minerva-style OLMES configuration (Lewkowycz et al., 2022); PopQA evaluates factual knowledge, especially long-tail entity knowledge (Mallen et al., 2023); and TruthfulQA evaluates whether models produce truthful answers rather than imitating common falsehoods (Lin et al., 2022). All tasks are evaluated using macro scores. We report the OLMES macro-aggregated primary score across the eight tasks as the final result. We use separate development and test sets for configuration selection and final evaluation, respectively. Details of the benchmark-specific split construction are provided in Appendix A.2.

The base model is evaluated with the same tokenizer and chat template used for SFT to avoid prompt-format mismatch.

[2] `allenai/Olmo-3-1025-7B`.

[3] `https://github.com/allenai/open-instruct`.

[4] `https://github.com/allenai/olmes`.

Ranking	Range Filter	Similarity Filter	Dev	Test
Random	–	–	53.61	51.92
RefNLL	–	–	52.41	50.73
Random	$\ell_{F_b} \leq 3.0$	$\gamma = 0.85$	54.22	52.08
RefNLL	$\ell_{F_b} \leq 3.0$	$\gamma = 0.85$	**54.32**	**53.02**
Full-data reference				
Full-data SFT	–	–	55.20	53.32

Table 1: Main results under the 10% total-training-token budget. The combined filter configuration is selected based on development performance, and the corresponding test results are reported without further configuration selection. The filtered Random row is a matched control using the same filter configurations. The unfiltered Random row is averaged over three data-selection seeds.

4.2 Main Results

Table 1 reports the main results under the 10% total-training-token budget. The unfiltered random baseline remains a strong baseline: on the development sets, it achieves a macro score of 53.61, compared with 52.41 for direct RefNLL ranking.

The same tendency is observed on the held-out test sets, where RefNLL ranking also underperforms random selection. This is consistent with recent observations that instruction data selection methods can struggle to consistently outperform random baselines across settings.

RefNLL ranking alone is informative but insufficient. The RefNLL top-ranked subset performs better than the corresponding tail-ranked subset, but still falls below random selection. This suggests that RefNLL captures useful prioritization information, while ranking alone does not define an effective token-budgeted subset.

We next consider the full role-separated configuration, which combines RefNLL prioritization with BaseNLL compatibility control and response-side cosine redundancy control. Based on development performance, we select $\ell_{F_b} \leq 3.0$ with $\gamma = 0.85$. This configuration reaches a development macro score of 54.32, improving over both direct RefNLL ranking and the unfiltered random baseline. The same configuration obtains a test macro score of 53.02, compared with 51.92 for random selection and 50.73 for direct RefNLL ranking. It also numerically exceeds the matched random control using the same filters, while approaching the 53.32 obtained by full-data SFT. Thus, the combination of prioritization, compatibility control, and redundancy control improves token-budgeted subset construction while using only 10% of the full training-token budget.

To better understand the contribution of the two control mechanisms, we next conduct component ablations in Section 5. These analyses examine whether compatibility control and redundancy control each contribute to the improvement over direct RefNLL ranking.

5 Analysis

The main results show that RefNLL provides a meaningful ranking, but the RefNLL top rank alone is not sufficient to form an effective subset under a token budget. We now analyze how each component of the role-separated subset construction framework contributes to the final performance.

5.1 Ranking-only selection

Before analyzing the effects of range filtering and redundancy filtering, we first examine whether likelihood-derived rankings are sufficient by themselves. Besides $s_{\mathrm{RefNLL}}(z)$, we also evaluate two NLL-difference-based scores as ranking signals:

$$s_{\mathrm{NLLDiff}}(z) = \ell_{F_b}(z) - \ell_{F_r}(z),$$

$$s_{\mathrm{RelNLLDiff}}(z) = \frac{\ell_{F_b}(z) - \ell_{F_r}(z)}{\ell_{F_b}(z)}$$

where F_b denotes the base model. s_{NLLDiff} measures the unnormalized NLL difference from the base model to the reference model, and $s_{\mathrm{RelNLLDiff}}$ refers to the relative difference. Examples with larger differences are ranked higher under both NLL-difference-based scores. The intuition is that a larger difference suggests that the example may be difficult for the base model but more predictable for the reference model, and may therefore be more informative under a limited SFT budget.

These two ranking signals serve two purposes in our study. First, they test whether cross-entropy-difference-style scores remain useful in our large-scale SFT setting. Second, they provide candidate traversal ranks on top of which more structured subset-construction rules can be applied.

Table 2 compares several ranking-only selections under the same 10% total-training-token budget. We include NLLDiff scores inspired by cross-entropy-difference data selection, where the score measures the difference in supervised-token NLL from the base model to a reference model. In the same-model variant, the reference model is the base model after full-data SFT; in the cross-model variant, the reference model is an external instruction-tuned model. We also evaluate RelNLLDiff, which normalizes this difference by BaseNLL.

The same-model NLLDiff setting is useful as a diagnostic baseline, because it measures how much

Ranking Signal	**Order**	**Dev Macro**
Random avg.	–	**53.61**
NLLDiff (same-model)	tail	52.89
	top	52.63
NLLDiff (cross-model)	tail	50.79
	top	44.84
RelNLLDiff (cross-model)	tail	50.51
	top	51.67
RefNLL	tail	51.13
	top	52.41

Table 2: Ranking-only selection under the 10% total-training-token budget. For the same-model variant, both NLL terms are computed with the same model before and after instruction tuning. For the cross-model variant, the reference model is an external instruction-tuned model, different from the base model.

Ranking	Range Filter	Dev Macro
Random avg.	–	53.61
RefNLL	–	52.41
RefNLL	$0.05 \leq \ell_{F_b} \leq 3.0$	54.24
	$0.20 \leq \ell_{F_b} \leq 3.0$	53.81
RefNLL	$\ell_{F_b} \geq 0.20$	51.53
	$\ell_{F_b} \leq 2.5$	53.58
	$\ell_{F_b} \leq 3.0$	54.24
	$\ell_{F_b} \leq 4.0$	53.81

Table 3: Effect of BaseNLL range filtering under the 10% total-training-token budget. Performance is reported for both one-sided and two-sided range filters.

a model trained on the full pool reduces loss on each example. However, it is less useful as a practical selection method, since it requires a model fine-tuned on the full data pool before selection. Cross-model NLLDiff avoids this circularity by using an external reference model, but it can introduce calibration issues because the two NLLs are computed by different models and potentially different tokenizers. RefNLL avoids taking a direct cross-model difference and uses the external reference model only to define a ranking.

The results show that none of the ranking-only variants clearly outperforms random selection. Among the practical variants, RefNLL gives the highest ranking-only score: under the 10% total-training-token budget, RefNLL Top exceeds RefNLL Tail, but still falls below the average random baseline. We therefore use RefNLL as the main ranking signal in the rest of the analysis, and treat the conversion from this ranking into a token-budgeted subset as a separate problem.

5.2 Effect of BaseNLL Range Filtering

We next analyze the effect of BaseNLL range filtering. Table 3 reports variants that keep the same ascending RefNLL rank but restrict the eligible examples by their BaseNLL ℓ_{F_b}. We consider both one-sided filters, which remove only one extreme of the BaseNLL distribution, and two-sided filters, which restrict the candidate pool to a moderate BaseNLL range.

The thresholds are chosen from the empirical distribution of ℓ_{F_b} in the full data pool. For example, the upper tail of the distribution is around $\ell_{F_b} \approx 3$, so we use $\ell_{F_b} \leq 3.0$ as a natural starting point for excluding very high-NLL examples. We use absolute NLL thresholds rather than fixed-percentile removal because our goal is not to discard a fixed amount of data, but to identify candidate regions that appear useful under a compact token budget.

Score	p1	p5	p10	p25	p50	p75	p90	p95
BaseNLL	0.15	0.21	0.25	0.39	0.88	1.45	2.23	3.13

Table 4: Empirical distribution of supervised-token average NLL scores on the SFT data pool. P_k denotes the k-th percentile. These statistics motivate the absolute BaseNLL thresholds used in Table 3, such as $\ell_{F_b} \approx 0.2$ near the lower tail and $\ell_{F_b} \approx 3.0$ near the upper tail.

Ranking	Similarity Filter	Dev Macro
Random avg.	–	53.61
RefNLL	–	52.41
RefNLL	$\gamma = 0.80$	54.19
	$\gamma = 0.85$	53.93
	$\gamma = 0.90$	53.98
	$\gamma = 0.95$	54.19

Table 5: Effect of response-side cosine filtering under the 10% total-training-token budget across various similarity thresholds γ.

The results of one-sided range filtering suggest that retaining the high-NLL tail is more harmful than retaining the low-NLL tail in this setting. Filtering high-NLL examples improves the development macro score of direct RefNLL ranking from 52.41 to 53.58, 54.24, and 53.81 with upper thresholds of $\ell_{F_b} \leq 2.5$, $\ell_{F_b} \leq 3.0$, and $\ell_{F_b} \leq 4.0$, respectively. In contrast, filtering only very low-NLL examples with $\ell_{F_b} \geq 0.20$ decreases performance to 51.53. This suggests that very high BaseNLL examples are less cost-effective under the 10% total-training-token budget, while low-BaseNLL examples are not simply useless or harmful.

The reported two-sided ranges show a similar tendency. The range $0.05 \leq \ell_{F_b} \leq 3.0$ reaches a development macro score of 54.24, while $0.20 \leq \ell_{F_b} \leq 3.0$ reaches 53.81, both improving over direct RefNLL ranking at 52.41.

Overall, these results indicate that restricting the candidate pool by BaseNLL can improve RefNLL-based subset construction, supporting its role as a compatibility-control mechanism in our framework.

5.3 Effect of Redundancy Filtering

We then examine whether direct RefNLL ranking suffers from redundancy in the selected subset. Since SFT loss is computed over assistant responses, we measure similarity using embeddings of the response side of each example rather than the

Ranking	Range Filter	Sim. Filter	Dev Macro
Random avg.	–	–	53.61
RefNLL	–	–	52.41
Random	$\ell_{F_b} \leq 3.0$	$\gamma = 0.85$	54.22
RefNLL	$\ell_{F_b} \leq 3.0$	$\gamma = 0.80$	54.16
	$\ell_{F_b} \leq 3.0$	$\gamma = 0.85$	**54.32**
	$\ell_{F_b} \leq 3.0$	$\gamma = 0.90$	53.76
	$\ell_{F_b} \leq 3.0$	$\gamma = 0.95$	53.72

Table 6: Combined effect of BaseNLL range filtering and response-side cosine filtering under the 10% total-training-token budget.

full instruction–response sequence. During traversal of the RefNLL ranking, an example is dropped if its response embedding is too similar to any already selected example.

Table 5 shows that response-side cosine filtering consistently improves over RefNLL-only selection. The macro score of direct RefNLL ranking reaches 52.41, while all tested cosine-filtering variants improve the score to 53.93–54.19. The improvement from cosine filtering indicates that redundancy control can make the RefNLL ranking more effective for token-budgeted subset construction.

This observation complements the BaseNLL analysis, which likewise shows that the RefNLL ranking benefits from additional control during subset construction. Cosine filtering does not change the ranking or introduce a new utility score; it changes how the ranked list is converted into a subset. Thus, the effectiveness of the RefNLL ranking depends not only on the ordering of examples but also on how the token budget is allocated during traversal.

5.4 Combining Range and Redundancy Constraints

After analyzing BaseNLL range filtering and response-side cosine filtering separately, we further examine their combined use. Table 6 reports the results of applying both filters during traversal of the same RefNLL ranking. The reported combined variants remain consistently strong, achieving development macro scores of 53.72–54.32, and all outperform direct RefNLL ranking at 52.41.

Among the reported combined configurations, $\ell_{F_b} \leq 3.0$ with $\gamma = 0.85$ achieves the highest development macro score of 54.32 and is therefore used as the full configuration in our main evaluation. Its gain over the corresponding standalone range-filter setting (54.24) and cosine-filter setting

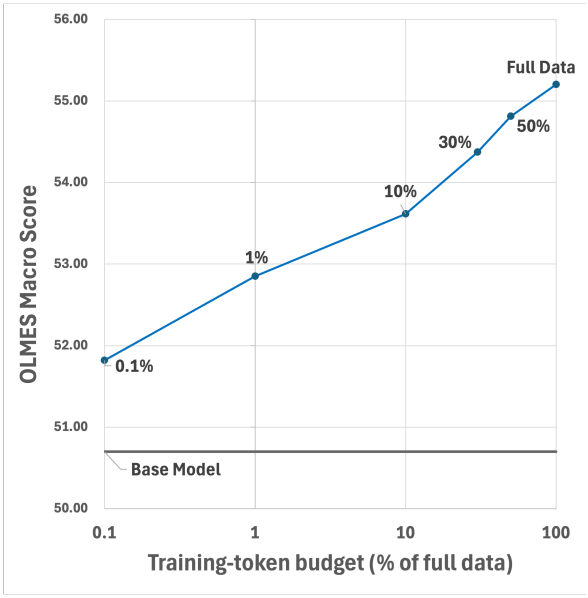


Figure 2: The OLMES macro performance on the development set under random selection across different total-training-token budgets. Reduced-budget results are averaged over three data-selection seeds. The pretrained base model and full-data SFT model are included as reference points.

(53.93) is modest, suggesting that the two controls are useful but not simply additive. Once the candidate region is restricted by BaseNLL, the redundancy structure encountered by the RefNLL traversal also changes; conversely, cosine filtering can change which BaseNLL regions are actually selected under the token budget.

On the test sets, this development-selected full configuration reaches 53.02, outperforming both the unfiltered random baseline at 51.92 and the matched random control at 52.08, while approaching full-data SFT at 53.32 (Table 1). These results reinforce the main conclusion of our analysis: likelihood-derived ranking is useful, but its effectiveness depends on how the ranked list is constrained and traversed.

5.5 Effect of Training-Token Budget

To further examine how random-selection performance changes with the available training budget, we conduct an additional budget-scaling analysis using random subsets at 0.1%, 1%, 10%, 30%, and 50% of the full-data token budget.

For each reduced-budget setting, we report the average over three random data-selection seeds. The pretrained base model and full-data SFT model are included as reference points. This analysis is intended to characterize the overall trend with re-

spect to training-token budget rather than to compare individual budget settings against the main experiments.

As shown in Figure 2, random-selection performance generally improves as more training tokens become available. The improvement is particularly pronounced in the low-budget regime, while the gains become smaller at larger budgets as performance approaches full-data SFT. Overall, random-selection performance increases with the available training-token budget in this setting.

5.6 Subset Size and Training Cost

A fixed token budget does not imply a fixed number of training examples. Under the 10% total-training-token budget, the random subsets contain approximately 86.1K examples on average, whereas the development-selected full-method subset contains 58.3K examples, despite both containing approximately 57.87M serialized training tokens. Correspondingly, the average example length increases from approximately 672 tokens for random selection to 993 tokens for the selected subset. The selected subset also contains a larger proportion of supervised response tokens (64.6% versus 60.7%).

This difference in subset composition also affects SFT training cost under our example-based batching setup. For reference, full-data SFT requires 13,466 optimization steps and approximately 44.5 hours of training. Under the 10% token budget, the random subsets require 1,340–1,356 optimization steps and approximately 4.4–4.5 hours of training, whereas the selected subset requires 912 steps and approximately 2.5 hours. These measurements cover SFT training only and do not include the offline scoring and subset-construction costs. Detailed subset statistics are provided in Appendix B.

6 Conclusion

This work revisits SFT data selection from the perspective of token-budgeted subset construction. Within the studied OLMo setting, we observe the strong performance of the random baseline. We apply an NLL-based ranking method, and the results suggest that the NLL rankings are informative but limited: direct top-ranked selection according to the RefNLL score underperforms the average random baseline.

To address this limitation, we proposed range filtering and similarity filtering from the perspectives of compatibility control and redundancy control. The experimental results show that these filters improve RefNLL-based token-budgeted subset construction in the studied OLMo setting. These results suggest that likelihood-derived signals are useful for prioritization and compatibility control, while embedding-based similarity provides complementary redundancy control.

The specific filters explored in this study are not intended to exhaust all possible instantiations of the role-separated framework. BaseNLL range filtering is one way to control base model suitability, and response-side cosine filtering is one way to control redundancy. Other views could be incorporated into the same framework, and we leave such extensions to future work.

Limitations

Our experiments are limited to the OLMo ecosystem, so the generality of the observed trends should be further tested on other model families and instruction mixtures. Most non-random selection results are reported from a single training run, so small differences among top-performing variants should not be interpreted as statistically significant. The 10% subset does not clearly surpass full-data training, suggesting room for stronger subset construction methods. The additional budget-scaling analysis in Section 5.5 considers random selection only; systematically evaluating the proposed selection strategy across multiple token budgets remains future work.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP26H02511, by the "R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models" project of the Ministry of Education, Culture, Sports, Science and Technology, and by JST Moonshot R&D Grant Number JPMJMS2011-35 (fundamental research). This work was partly achieved through the use of the Supercomputer system at the Information Initiative Center, Hokkaido University, Sapporo, Japan.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 355–362.

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

Harshita Diddee and Daphne Ippolito. 2025. Chasing random: Instruction selection strategies fail to generalize. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1943–1957.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. 2025. OLMES: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5020–5048, Albuquerque, New Mexico. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring Mathematical Problem Solving With the MATH Dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. 2025. Large-scale data selection for instruction tuning. *arXiv preprint arXiv:2503.01807*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tülu 3: Pushing Frontiers in Open Language Model Post-Training. In *Second Conference on Language Modeling*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving Quantitative Reasoning Problems with Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857. Curran Associates, Inc.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024. SelectIT: Selective Instruction Tuning for LLMs via Uncertainty-Aware Self-Reflection. In *Advances in Neural Information Processing Systems*, volume 37, pages 97800–97825. Curran Associates, Inc.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 9802–9822.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Findings of ACL, pages 13003–13051. Association for Computational Linguistics.

Jun Suzuki, Heiga Zen, and Hideto Kazawa. 2023. Extracting representative subset from extensive text data

for training pre-trained language models. *Information Processing & Management*, 60(3):103249.

Team OLMo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee F. Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. OLMo 3. *CoRR*, abs/2512.13961.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, and 23 others. 2025. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. In *Advances in Neural Information Processing Systems*, volume 36, pages 74764–74786. Curran Associates, Inc.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. In *International Conference on Learning Representations*.

Yang Wu, Huayi Zhang, Yizheng Jiao, Lin Ma, Xiaozhong Liu, Jinhong Yu, Dongyu Zhang, Dezhi Yu, and Wei Xu. 2025. ROSE: A reward-oriented data selection framework for LLM task-specific instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13200–13219, Suzhou, China. Association for Computational Linguistics.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54104–54132. PMLR.

Tingyu Xia, Bowen Yu, Kai Dang, An Yang, Yuan Wu, Yuan Tian, Yi Chang, and Junyang Lin. 2025. Rethinking data selection at scale: Random selection is almost all you need. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 2698–2711, Suzhou, China. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. LIMO: Less Is More for Reasoning. In *Second Conference on Language Modeling*.

Jia Zhang, Chen-Xi Zhang, Yao Liu, Yi-Xuan Jin, Xiao-Wen Yang, Bo Zheng, Yi Liu, and Lan-Zhe Guo. 2025a. D3: Diversity, Difficulty, and Dependability-Aware Data Selection for Sample-Efficient LLM Instruction Tuning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 8348–8356. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, and 4 others. 2026. A Survey of Large Language Models. *Frontiers Comput. Sci.*, 20(12):2012627.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: Less Is More for Alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021. Curran Associates, Inc.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Item	Value
Base model	`allenai/Olmo-3-1025-7B`
Tokenizer	`allenai/Olmo-3-1025-7B`
Training pipeline	Open-Instruct SFT pipeline
Mixed precision	`bf16`
Distributed training	DeepSpeed ZeRO-3
Number of GPUs	4 H100 GPUs
Maximum sequence length	4096
Per-device train batch size	4
Gradient accumulation steps	8
Learning rate	2×10^{-5}
Learning-rate scheduler	Linear
Warmup ratio	0.03
Weight decay	0.0
Number of training epochs	2
Training seed	8
Chat template	`olmo`
Flash attention	Enabled
Gradient checkpointing	Enabled

Table 7: Training configuration used for the SFT experiments.

Benchmark	OLMES task configuration
IFEval	`ifeval::tulu`
BBH	`bbh:cot-v1::olmes`
DROP	`drop::llama3`
MMLU	`mmlu:cot::summarize`
GSM8K	`gsm8k::tulu`
MATH	`minerva_math::tulu`
PopQA	`popqa::tulu`
TruthfulQA	`truthfulqa::olmo1`

Table 8: OLMES task configurations used for evaluation.

A Experimental Details

A.1 Training Configuration

All SFT experiments are initialized from `allenai/Olmo-3-1025-7B`, and use the corresponding tokenizer. Unless otherwise stated, all runs use the same training configuration and differ only in the selected training subset. Table 7 summarizes the main training hyperparameters.

A.2 Evaluation Protocol

All checkpoints are evaluated with the same OLMES configuration. We compute mean macro score over the eight benchmark scores. Table 8 lists the OLMES task configurations. We use separate development and test sets for configuration selection and final evaluation, respectively. Table 11 summarizes the benchmark-specific split construction.

A.3 Artifact Licenses

We use publicly available artifacts in accordance with their associated licenses and terms of use. The SFT data pool is the Tulu-3 OLMo-2 SFT mixture, the base model is OLMo-3-1025-7B, and the reference and embedding models are from the Qwen family. We also use the Open-Instruct SFT pipeline and OLMES evaluation framework. We refer users to the corresponding model, dataset, and code repositories for the detailed license and usage terms.

Item	Value
Initial instances	866,138
Final instances	861,749
Filtered instances	4,389
Total training tokens	578,700,922
Trainable response tokens	351,557,528
Average tokens per instance	671.5

Table 9: Statistics of the full Tulu-3 OLMo-2 SFT mixture after preprocessing.

Subset	Examples	Total Tok.	Train Tok.
Random, seed 6	85.6K	57.87M	35.18M
Random, seed 8	86.7K	57.87M	35.13M
Random, seed 42	86.0K	57.87M	35.14M
Full method	58.3K	57.87M	37.40M

Table 10: Dataset statistics for representative 10% total-training-token budget subsets. "Train Tok." denotes trainable response tokens. "Full method" denotes RefNLL ranking with $\ell_{F_b} \leq 3.0$ and cosine filtering with $\gamma = 0.85$.

B Data Statistics

B.1 Full Data

Table 9 reports the post-processing statistics of the full Tulu-3 OLMo-2 SFT mixture used as the SFT data pool.

B.2 Selected Subsets

All token-budgeted subsets in the main experiments are constructed under a 10% total-training-token budget, corresponding to approximately 57.9M serialized training tokens. Because the budget is defined over total serialized tokens rather than the number of examples, different selection rules can result in different numbers of selected examples and different amounts of trainable response tokens.

C Development-Set Per-Benchmark Results

Table 12 reports development-set per-benchmark results for the RefNLL-based selection experiments and component ablations.

Benchmark	Development	Test
Official splits		
DROP	Official train split	Official validation split
MMLU	Official validation split	Official test split
MATH	Official train split	Official test split
GSM8K	Official train split	Official test split
Constructed splits		
IFEval	The official train split is divided approximately equally into development and test sets.	
PopQA	The data are divided approximately equally based on `prob_id`, while samples sharing the same `subj_id` are kept in the same split.	
TruthfulQA	Samples are divided approximately equally within each category to keep the category distributions balanced across the two splits.	
BBH	Each of the 27 subtasks is divided approximately equally, and the resulting halves are combined into development and test sets.	

Table 11: Development/test split construction used for configuration selection and final evaluation. Official splits are reused where appropriate; otherwise, benchmark-specific split rules are applied.

Model / setting	IFEval	BBH	DROP	MMLU	GSM8K	MATH	PopQA	TruthfulQA	Dev Macro
Reference results									
Base model	37.04	40.35	62.51	64.53	83.58	40.57	21.52	55.54	50.70
Full-data SFT	68.15	39.71	60.80	64.89	86.03	48.40	19.76	53.88	55.20
Random baseline variance									
Random, seed 6	61.48	40.35	62.28	64.29	81.32	47.16	21.81	54.03	54.09
Random, seed 8	54.07	39.32	62.20	65.64	81.13	46.44	21.15	52.95	52.86
Random, seed 42	61.48	40.83	61.38	67.05	82.10	45.75	21.35	51.17	53.89
Random avg.	59.01	40.16	61.95	65.66	81.52	46.45	21.44	52.72	53.61
Ranking-only baselines									
RefNLL(top)	54.07	40.08	62.19	62.66	84.73	46.41	20.78	48.38	52.41
RefNLL(tail)	52.22	41.94	59.49	64.64	81.69	39.91	20.03	49.13	51.13
BaseNLL range filtering									
RefNLL(top), $0.05 \leq \ell_{F_b} \leq 3.0$	64.07	42.14	63.17	65.81	83.18	46.08	21.02	48.42	54.24
RefNLL(top), $0.20 \leq \ell_{F_b} \leq 3.0$	59.26	42.83	61.84	65.54	85.19	46.23	21.20	48.38	53.81
RefNLL(top), $\ell_{F_b} \leq 2.5$	59.26	42.17	62.91	65.29	83.97	45.20	20.87	48.98	53.58
RefNLL(top), $\ell_{F_b} \leq 3.0$	64.44	42.72	62.98	64.41	83.31	44.90	21.50	49.68	54.24
RefNLL(top), $\ell_{F_b} \leq 4.0$	59.63	42.69	61.61	65.47	84.88	45.21	20.39	50.59	53.81
RefNLL(top), $\ell_{F_b} \geq 0.20$	54.07	38.49	60.82	55.95	85.24	44.28	20.37	52.98	51.53
Response-side cosine filtering									
RefNLL(top), $\gamma = 0.80$	63.33	43.04	62.37	63.22	84.92	46.27	20.63	49.74	54.19
RefNLL(top), $\gamma = 0.85$	62.59	43.78	62.47	62.64	84.02	46.27	20.82	48.84	53.93
RefNLL(top), $\gamma = 0.90$	60.37	42.37	61.77	65.95	83.94	46.54	21.21	49.73	53.98
RefNLL(top), $\gamma = 0.95$	64.81	42.55	62.58	64.97	83.17	46.89	20.37	48.19	54.19
Combined BaseNLL and cosine filtering									
Random, $\ell_{F_b} \leq 3.0$, $\gamma = 0.85$	62.47	40.98	62.43	65.21	81.07	46.93	21.47	53.17	54.22
RefNLL(top), $\ell_{F_b} \leq 3.0$, $\gamma = 0.80$	61.48	42.97	63.06	63.92	84.12	46.58	21.42	49.77	54.16
RefNLL(top), $\ell_{F_b} \leq 3.0$, $\gamma = 0.85$	63.70	42.96	62.53	63.63	83.89	46.84	21.00	49.98	54.32
RefNLL(top), $\ell_{F_b} \leq 3.0$, $\gamma = 0.90$	58.15	43.39	62.59	63.87	83.86	46.41	21.35	50.50	53.76
RefNLL(top), $\ell_{F_b} \leq 3.0$, $\gamma = 0.95$	57.78	42.61	62.38	64.40	84.79	46.78	20.94	50.10	53.72

Table 12: Development-set per-benchmark results. Reduced-data selection rows use the 10% total-training-token budget; the base model and full-data SFT are included as reference points.