ATCS: ACCELERATING TARGET-MODEL-AWARE CORESET SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale supervised fine-tuning (SFT) of foundation models increasingly relies on instruction data pools aggregated from heterogeneous sources such as the web, open communities, and synthetic generation pipelines. To support reliable model adaptation, such data must be transformed into a compact, high-value training set by removing redundant, noisy, and low-utility examples. This transformation naturally calls for data selection methods that can reduce the training pool while preserving the information needed for downstream adaptations. Target-model-aware coreset selection is particularly effective for this purpose because it evaluates candidates according to their utility for the model to be fine-tuned. However, existing target-model-aware methods are difficult to deploy at scale. Specifically, they typically require the target model to score every candidate instruction-response pair, turning data selection itself into a costly inference workload. To address this bottleneck, we propose ATCS, a general acceleration framework that treats selection as a two-stage data processing pipeline. ATCS exploits the agreement between the proxy and target models in identifying low-value samples and salient instructional content, reducing both the number and length of records evaluated by the target model. Besides, ATCS is compatible with multiple utility functions and supports cross-utility selection across coarse filtering and target-model reranking. Experiments on large-scale instruction-tuning datasets show that ATCS preserves downstream performance comparable to exhaustive target-model-aware selection while reducing end-to-end selection time by more than 77%, making model-aware coreset selection practical for scalable LLM data preparation.

1 INTRODUCTION

Large-scale supervised fine-tuning (SFT) (Ouyang et al., 2022) has become a standard approach for adapting pretrained large language models to downstream applications. In practice, modern instruction-tuning pipelines often begin with large candidate pools aggregated from heterogeneous sources, including the web, open communities, human annotations, and synthetic generation. These pools contain not only valuable training signals, but also substantial redundancy, templated variants, inconsistent annotations, and low-utility examples (Ratner et al., 2019; Varma & Ré, 2018; Krishnan et al., 2016). Using these pools directly can waste computation on redundant or uninformative examples. Therefore, selecting a compact, high-value training subset can retain useful adaptation signals while reducing such overhead. This data preparation step affects both the computational cost and downstream performance of SFT, making scalable selection increasingly important for practical LLM adaptation.

Coreset selection addresses this challenge by identifying a compact training subset from a large candidate pool. Under a fixed training budget, it aims to preserve the most useful training signals for downstream SFT. An effective coreset can reduce training cost, shorten fine-tuning cycles, and improve the accessibility of LLM adaptation in resource-constrained environments. More importantly, coreset selection turns instruction tuning from a passive data accumulation process into an active data curation pipeline. The system decides which records should be retained, which should be discarded, and how to allocate limited training budget across a heterogeneous data pool.

However, existing coreset selection methods face a tension between efficiency and effectiveness. Model-agnostic or representation-based methods (Abbas et al., 2023; Sener & Savarese, 2018;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Bachem et al., 2018) rely on heuristic rules, deduplication, clustering, or embedding-space coverage to remove redundant or low-quality samples. While computationally efficient and easy to deploy at scale, these approaches lack direct interaction with the target LLM and may therefore miss samples that are truly informative for the model being adapted. In contrast, target-model-aware methods estimate candidate utility by querying the target model itself. Since their scores reflect how the target model responds to each sample, these methods can better identify examples that are informative for the specific model being adapted, and therefore often preserve downstream performance better under aggressive data reduction. Nevertheless, this model awareness comes at the cost of substantial computational overhead. Each utility score requires a target-model forward pass over an instruction-response sequence, making exhaustive scoring prohibitively costly for massive candidate pools and billion-parameter models. Proxy-based methods such as Superfiltering (Li et al., 2024a) reduce this cost by using a smaller model for scoring. Although effective, relying entirely on proxy scores may miss selection preferences specific to the target model, leaving room for improvement through target-model evaluation.

This creates a practical bottleneck for large-scale SFT. Exhaustive target-model scoring can require substantial computation before training begins, while replacing it entirely with proxy scoring sacrifices direct target-model judgment. The challenge becomes more pronounced when candidate pools are large and individual sequences are long. Therefore, a key question arises: *can the benefits of target-model-aware selection be retained while substantially reducing the computation required to construct the coreset?*

In this work, we argue that the scalability of target-model-aware coreset selection should be improved at the pipeline level. Simply designing another utility function does not resolve the core bottleneck if the target model still has to scan the entire candidate pool with long instruction-response inputs. Instead, a scalable solution should reduce unnecessary target-model computation along two dimensions: (i) the number of candidate records evaluated by the target model, and (ii) the amount of text processed during each target-model scoring operation.

Guided by this principle, we propose ATCS[1], a general acceleration framework for target-model-aware coreset selection. Our key insight is that proxy computation can guide both candidate selection and the choice of content used for target-model scoring. Proxy utility scores indicate which candidates warrant further evaluation, while response-conditioned attention identifies relevant instructional content within each candidate. ATCS couples these decisions by reusing the same proxy forward pass for candidate filtering and sentence localization. The proxy thus goes beyond sample ranking to help construct shorter scoring inputs whose retained content is guided by relevance to the reference response. Two empirical observations support this design. First, our analysis shows substantial agreement between proxy and target models in identifying samples with low utility scores. Second, their response-conditioned attention often highlights similar instructional sentences. These observations motivate using the proxy to narrow the candidate pool and shorten its inputs without requiring it to reproduce the target model's complete ranking. The target model then performs the final utility-based selection among the retained candidates using shortened inputs. After selection, the full original examples are recovered for SFT.

Notably, ATCS is a general acceleration layer rather than a new utility function, so it can be combined with different target-model-aware scoring metrics. Observing that each utility typically captures only one aspect of sample usefulness, we further extend ATCS to a cross-utility setting, where different utilities are used for coarse filtering and fine-grained reranking to combine complementary selection signals within the same scalable pipeline.

In summary, this work makes the following contributions:

- We propose ATCS, a general acceleration framework for target-model-aware coreset selection. ATCS filters out low-value samples through lightweight proxy-model scoring, localizes key instructional sentences using attention-based saliency, and adaptively truncates inputs according to their retained information, thereby reducing the computational cost of coreset selection while largely preserving coreset quality.

[1]The source code is available in an anonymous repository.

- We investigate the limitations of selection under a single utility and explore cross-utility selection within ATCS. Our analysis shows that combining utilities across stages can improve downstream performance and reduce coreset redundancy in the evaluated settings.
- We conduct extensive experiments on three large-scale instruction-tuning datasets, multiple target and proxy models, and six downstream benchmarks. ATCS achieves downstream performance broadly comparable to exhaustive target-model scoring, while reducing end-to-end selection time by more than 77%.

2 Related Work

Target-independent instruction-data selection. Instruction-tuning data can be selected according to quality, diversity, and representativeness without explicitly evaluating its utility to the target model. LIMA (Zhou et al., 2023a) demonstrates that a small, carefully curated dataset can support effective alignment, although expert curation is difficult to scale. Representation-based methods promote diversity or remove near-duplicates through embedding-space clustering and coverage objectives (Abbas et al., 2023; Sener & Savarese, 2018). For instruction tuning, DEITA (Liu et al., 2024b) combines estimated data quality, complexity, and diversity, while InsTag (Lu et al., 2024) characterizes instructions using semantic tags to improve task coverage. LLM-as-a-judge methods such as ALPAGASUS (Chen et al., 2023) instead use an external model to assess instruction quality. These approaches improve data efficiency, but their selection criteria do not directly reflect the knowledge state of the model to be fine-tuned.

Efficient coreset selection. Several methods reduce selection cost through proxy models or compact sample representations. Superfiltering (Li et al., 2024a) uses a weak language model to approximate instruction-following difficulty and directly select training examples for substantially larger models. SCAR (Li et al., 2025) uses lightweight representations to rank responses according to style consistency. FastCore (Chai et al., 2023) accelerates gradient-based coreset construction through clustering and distance approximation, whereas TAGCOS (Zhang et al., 2025) clusters gradient representations to select diverse and representative instruction subsets. These methods improve scalability by replacing expensive evaluation with proxy signals or representation-space approximation.

Target-model-aware selection. Target-model-aware methods evaluate samples according to signals obtained from the model being fine-tuned. IFD (Li et al., 2024b) measures instruction-following difficulty by comparing response likelihoods with and without the instruction, while UFS (Zhang et al., 2024) estimates the target model's familiarity with individual examples. SelectIT (Liu et al., 2024a) uses uncertainty-aware self-reflection to assess instruction quality, and LESS (Xia et al., 2024) estimates the influence of training examples on target tasks. Such signals better reflect model-specific data value, but applying them over the complete candidate pool requires extensive target-model inference or gradient computation.

Existing approaches leave an unresolved trade-off between efficiency and target-model awareness. Proxy-based methods avoid expensive target-model evaluation, but their final selections are determined entirely by proxy signals and may therefore miss target-specific preferences. Representation- and gradient-based methods emphasize feature-space coverage or gradient approximation rather than directly estimating training data value for the target model. Conversely, target-model-aware methods retain model-specific information but typically incur high computational costs. Accordingly, in this paper, we aim to preserve target-model-aware selection while substantially reducing its computational overhead.

3 Methodology

3.1 Problem Formulation and Analysis

We first formalize the *target-model-aware* coreset selection for supervised fine-tuning (SFT). Let the candidate dataset be $C = \{(x_i, y_i)\}_{i=1}^{|C|}$, where x_i denotes the instruction of the i-th sample, y_i denotes the corresponding reference response, and $|C|$ is the total number of candidate samples. Let $\mathcal{M}$ denote the target large language model (LLM) to be fine-tuned.

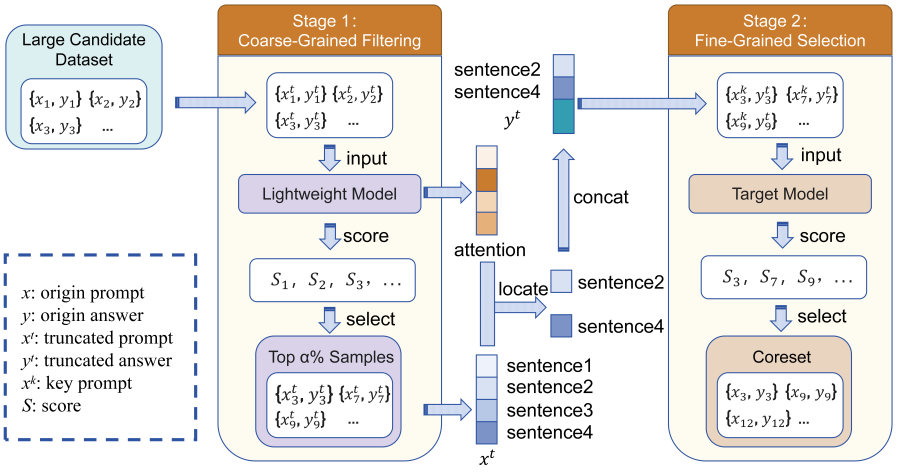


Figure 1: Overview of the proposed two-stage acceleration framework. **Stage 1 (Coarse-Grained Filtering)** uses a lightweight model to evaluate truncated samples and remove low-value candidates. **Stage 2 (Fine-Grained Selection)** applies key-sentence localization and then uses the target model to score only refined high-potential samples, producing the final coreset for SFT.

Coreset selection aims to choose a subset $S \subseteq C$ with a fixed budget $|S| = K$ such that fine-tuning on S yields the best downstream performance. Formally, the ideal objective can be written as

$$S^* = \arg \max_{S \subseteq C,\ |S|=K} \mathrm{Perf}\big(\mathrm{SFT}(\mathcal{M}, S)\big), \tag{1}$$

where $\mathrm{SFT}(\mathcal{M}, S)$ denotes fine-tuning $\mathcal{M}$ on subset S, and $\mathrm{Perf}(\cdot)$ denotes downstream evaluation after fine-tuning.

However, directly optimizing Eq. (1) is infeasible in practice, because evaluating even a single candidate subset requires running SFT and downstream testing, which is prohibitively expensive over large candidate pools. Therefore, existing target-model-aware methods introduce a per-sample utility function $s(\mathcal{M}, x_i, y_i)$ to estimate how useful a sample is for improving the target model. Typical utilities include negative log-likelihood (NLL), instruction-following difficulty (IFD), and unfamiliar score (UFS). Under this surrogate view, subset selection is approximated by

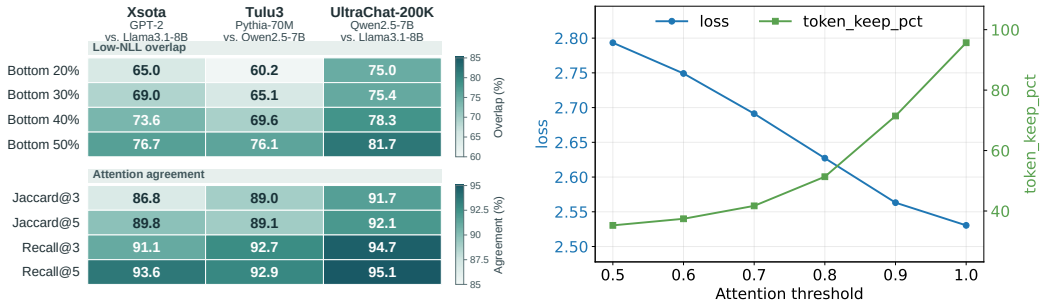
$$\max_{S \subseteq C,\ |S|=K} \sum_{(x_i, y_i) \in S} s(\mathcal{M}, x_i, y_i). \tag{2}$$

Eq. (2) is the practical objective optimized by most utility-based target-model-aware selection methods. Importantly, it is still only a surrogate for the true downstream objective in Eq. (1). In particular, downstream performance may depend not only on the individual utility of selected samples, but also on structural properties of the selected subset, such as redundancy and semantic coverage. We return to this point in Section 4.2, where we discuss why exact utility maximization does not always imply the best downstream subset.

Although utility-based target-model-aware selection is often effective, its computational cost is extremely high at scale because the target model must score a massive number of instruction-response pairs. In the common teacher-forcing setting used by scoring functions, the model evaluates the response y conditioned on the instruction x and the preceding response tokens. Either x or y can be very long in real-world instruction datasets, further increasing the cost of each scoring pass.

3.2 ATCS: Two-Stage Acceleration with Key-Sentence Localization

To address the bottleneck above, we propose **ATCS** (Accelerating Target-model-aware Coreset Selection), a two-stage framework that uses proxy signals to reduce both the number of candi-



(a) Cross-model agreement.

(b) Effect of the attention threshold.

Figure 2: Cross-model agreement and information retention.

dates evaluated by the target model and their scoring length. ATCS couples candidate filtering with response-conditioned input shortening through a shared proxy forward pass. Filtering alone reduces the number of target-model evaluations but leaves the cost of processing each retained sequence. ATCS therefore reuses the attention produced during proxy scoring to determine which instructional sentences to retain for those evaluations. The same proxy computation supports both sample-level filtering and sentence-level content selection, without a separate forward pass for localization.

As illustrated in Figure 1, stage 1 performs *coarse-grained filtering* with a lightweight proxy model to narrow the full candidate pool to a high-potential subset while collecting response-to-instruction attention. Stage 2 performs *fine-grained selection* with the target model on the retained candidates, using shorter instructions reconstructed from the attention-guided key sentences. A cumulative attention threshold determines which sentences to retain, allowing the scoring length to adapt to each sample's attention distribution. These sentences are restored to their original order and combined with the truncated reference response for target-model scoring.

Stage 1: Coarse-Grained Filtering with a Lightweight Model

Stage 1 aims to reduce the number of samples requiring expensive target-model evaluation. Given a candidate pool C, a lightweight proxy model estimates the utility of each sample, after which only the top $\alpha|C|$ candidates are retained for Stage 2.

Coarse filtering does not require the proxy model to reproduce the complete utility ranking of the target model. Instead, it requires sufficient agreement in identifying samples with clearly low utility. To examine this property, we independently construct the bottom-q utility subsets using the two models. As shown in Figure 2a, across all three model pairs, the overlap between their bottom-utility subsets ranges from 60.2% to 81.7% for thresholds of 20–50%, consistently exceeding the corresponding random expectations of 20–50%. This agreement demonstrates that proxy scoring provides a reliable approximation for identifying low-utility samples prior to target-model evaluation, supporting its role as an efficient preliminary filtering mechanism.

Since lightweight proxy models typically support shorter contexts than modern target LLMs, we construct a length-controlled version of each sample for coarse scoring. Conversations are truncated at turn boundaries whenever possible to preserve a coherent context. We additionally use only the first T_y response tokens for utility estimation, allocating as much of the limited context budget as possible to the instruction. The proxy model then scores these length-controlled samples, and the candidate pool is ranked according to the resulting utility values.

During proxy-based utility scoring, we extract response-to-instruction attention from the same forward pass and aggregate it into sentence-level saliency scores, without requiring a separate forward pass. Figure 2a shows that the top-attended instruction sentences identified by the two models achieve Jaccard scores of 0.868–0.917 and recall scores of 0.911–0.947 across datasets, model families, and model scales. Proxy attention therefore provides an inexpensive signal for locating salient instruction content. The retained candidates and their sentence-level saliency scores are subsequently passed to Stage 2 for target-model reranking with key-sentence-refined inputs.

Stage 2: Fine-Grained Selection with the Target Model

Stage 2 applies the target model only to the reduced candidate set returned by Stage 1. To further reduce the target-model inference cost, we shorten each retained instruction by reconstructing it from the most informative sentences identified by the attention signal collected in Stage 1.

During Stage 1, the lightweight model produces attention weights over the instruction sequence during its forward pass. We reuse these attention weights to estimate which instruction sentences are most relevant to the response. Specifically, we extract the attention weights from the last Transformer layer of the lightweight model and average them over attention heads. Let $A \in \mathbb{R}^{(N_x+N_y)\times(N_x+N_y)}$ denote the resulting attention matrix, where N_x and N_y are the token lengths of the instruction and the truncated response, respectively. We extract the response-to-instruction submatrix as $A^{y\to x} = A[N_x : N_x + N_y,\ 0 : N_x]$. For each instruction token x_j, we compute its response-conditioned attention score by averaging the attention it receives from all response tokens:

$$a_j = \frac{1}{N_y} \sum_{t=1}^{N_y} A_{t,j}^{y\to x}. \tag{3}$$

We then aggregate token-level scores into sentence-level scores. For a sentence s_m containing instruction-token indices $\mathcal{I}_m$, its score is the total attention mass assigned to its tokens:

$$r_m = \sum_{j\in\mathcal{I}_m} a_j. \tag{4}$$

We use summation rather than averaging because our objective is to retain sentences containing the greatest amount of response-relevant information. Averaging measures attention density and may underweight sentences whose relevant evidence is distributed across multiple tokens, whereas summation preserves their total contribution.

To select key sentences, we sort all sentences by r_m in descending order. Instead of retaining a fixed number of sentences or imposing a fixed token budget, we select the highest-scoring sentences until their cumulative normalized attention score reaches a threshold τ:

$$\frac{\sum_{s_m\in\mathcal{S}_{\text{keep}}} r_m}{\sum_{s_m\in\mathcal{S}} r_m} \geq \tau. \tag{5}$$

The selected sentences are then restored to their original order and concatenated to form the key instructional content for target-model utility estimation.

Figure 2b shows that lowering the attention threshold rapidly reduces the retained-token ratio while increasing response loss only modestly. At $\tau = 0.7$, the reconstructed instruction retains about 40% of the original tokens, while response NLL increases by only about 6%. This result indicates that the selected sentences preserve most of the response-relevant information despite substantial input shortening. Since self-attention cost grows quadratically with sequence length, the reduced input can substantially lower target-model computation in Stage 2.

For each retained sample, we construct a refined input consisting of: (i) key sentences selected from x_i by the cumulative attention threshold, and (ii) the same truncated response y_i used in Stage 1.

We then apply the target model to compute utility scores on the refined samples. Based on these scores, we select the top K samples as the final coreset. After selection, we recover the corresponding original full examples, and use these complete samples for subsequent SFT training.

3.3 Cross-Utility Selection

Existing utility-based methods usually perform exact top-K selection under a single surrogate utility objective. Formally, given a utility function $s(\cdot)$, they select

$$S^* = \arg\max_{S\subseteq C,\ |S|=K} \sum_{z\in S} s(z). \tag{6}$$

This objective is optimal with respect to the chosen utility score itself. However, it does not necessarily coincide with the true downstream objective. The reason is that downstream performance is affected not only by the individual utility scores of selected samples, but also by the composition of the selected subset, including diversity, semantic coverage, and task-specific data distribution.

In particular, optimizing a single utility may bias the selected coreset toward a narrow type of examples. For example, NLL, IFD, and UFS emphasize different aspects of sample usefulness, such as token-level likelihood, relative instruction-following difficulty, and uncertainty-related signals. If the selection process always maximizes only one of these utilities, it may over-concentrate on the samples favored by that criterion while overlooking samples that are useful under other criteria. This can reduce the diversity and coverage of the resulting coreset, and may lead to suboptimal downstream performance even when the selected samples have high utility scores.

Our experimental observations support this concern. First, as shown in Section 4.1, selecting the highest-scoring samples under a single target-model utility does not always yield the best downstream performance, suggesting that exact utility maximization is not necessarily aligned with the final SFT objective. Second, the redundancy analysis in Section 4.2 shows that exhaustive selection under a single utility can concentrate the coreset on similar samples. These findings motivate combining complementary utility signals to reduce reliance on a single selection criterion.

Motivated by this observation, we further consider a cross-utility variant of ATCS. Instead of requiring the same utility to be used in both stages, we allow Stage 1 and Stage 2 to adopt different utilities. This cross-utility design enables different utilities to play complementary roles. The Stage 1 utility controls which region of the candidate pool is preserved after low-cost filtering, while the Stage 2 utility determines the final target-model-aware ranking within that region. As a result, the framework can combine different notions of sample usefulness rather than being restricted to a single scoring criterion throughout the entire pipeline. In Section 4.2, we empirically evaluate different cross-utility combinations and show that they outperform homogeneous utility settings in most cases.

4 Experiments

4.1 Downstream Performance and Selection Cost

We evaluate ATCS on three instruction-tuning datasets: Xsota (Lu et al., 2024), Tulu3 (Lambert et al., 2024), and UltraChat-200K (HuggingFaceH4, 2023). Unless otherwise specified, each experiment selects a 10K coreset. We use Llama3.1-8B or Qwen2.5-7B/14B as the target model and GPT-2 or Pythia-70M as the proxy, as specified in Table 1. For this comparison, the coarse-stage retention ratio is $\alpha = 0.1$, the cumulative attention threshold is $\tau = 0.7$, and utility scoring uses at most 50 response tokens. We instantiate ATCS with three utilities: NLL, IFD (Li et al., 2024b), and UFS (Zhang et al., 2024). The baselines comprise two efficiency-oriented coreset selectors, Superfiltering (Li et al., 2024a) and FastCore (Chai et al., 2023), and two single-stage variants of ATCS: Coarse selects the coreset using only the proxy model, whereas Fine selects it through target-model scoring. After fine-tuning the selected coresets under the same LoRA protocol, we evaluate the target models on ARC-C (Clark et al., 2018), BBH (Suzgun et al., 2023), IFEval (Zhou et al., 2023b), and MMLU (Hendrycks et al., 2020) in the zero-shot setting; on GSM8K in the eight-shot setting; and on AlpacaEval using GPT-5.4 to judge each model against the Random baseline. Table 1 summarizes downstream performance across the experimental settings. Dataset descriptions, experimental details and additional results are provided in Appendix.

As shown in Table 1, coreset selection generally improves over Random, underscoring the value of selecting informative training examples. Among the efficient methods, ATCS remains the most competitive with exhaustive Fine selection overall. Superfiltering and Coarse rely on proxy-model signals. Although they reduce selection cost and yield useful coresets, the absence of target-model scoring leaves a performance gap relative to ATCS. FastCore accelerates gradient-based coreset selection by clustering samples in the feature space and selecting representatives to approximate the full-data gradient. Although it improves over Random, its lower performance than ATCS suggests that gradient approximation may not fully capture the utility of individual instruction–response pairs for target-model fine-tuning.

Figure 3a compares the selection time of ATCS and Fine under IFD. ATCS reduces end-to-end time by 77.2–85.8% while retaining target-model reranking. By reducing both the number of candidates scored by the target model and their scoring length, Stage 2 takes less than 10% of Fine's total selection time. Figure 3b further compares the 10K coresets selected by ATCS, Coarse, and Fine on Xsota, using Llama3.1-8B as the target model, GPT-2 as the proxy, and NLL as the selection utility. The 3,354 samples shared by Coarse and Fine indicate that the proxy already captures part of the

Table 1: Downstream performance of 10K coresets.

Method	ARC-C	BBH	IFEval	MMLU	GSM8K	AlpacaEval	Avg
Xsota / Llama3.1-8B / GPT-2 / UFS							
Random	75.89	60.33	35.27	53.22	49.65	50.00	54.06
Coarse	76.94	62.76	37.52	54.31	51.26	50.30	55.52
Fine	79.32	61.82	39.39	56.52	54.55	61.30	**58.82**
FastCore	77.66	61.13	39.64	53.60	53.32	55.60	56.83
Superfiltering	77.23	62.11	36.89	54.68	51.88	51.50	55.72
ATCS	77.97	63.36	38.30	55.79	54.21	60.40	58.34
Tulu3 / Qwen2.5-7B / GPT-2 / IFD							
Random	30.01	62.26	38.86	60.65	50.22	50.00	48.67
Coarse	28.20	66.44	40.80	50.50	51.56	51.90	48.23
Fine	29.49	61.57	33.59	32.37	55.89	46.20	43.19
FastCore	27.04	63.55	39.44	62.52	52.05	53.00	49.60
Superfiltering	27.99	65.85	40.24	50.21	53.22	52.50	48.34
ATCS	26.44	65.53	40.36	69.42	56.68	57.80	**52.71**
UltraChat-200K / Llama3.1-8B / GPT-2 / NLL							
Random	75.47	60.36	36.98	52.02	49.87	50.00	54.12
Coarse	77.63	62.57	36.63	51.40	51.78	52.00	55.34
Fine	79.66	62.19	41.81	57.16	53.83	60.10	**59.13**
FastCore	77.84	62.33	37.03	52.56	52.33	53.20	55.88
Superfiltering	77.84	61.98	36.22	52.65	52.14	53.50	55.72
ATCS	79.97	63.76	39.64	55.75	54.92	60.40	59.07

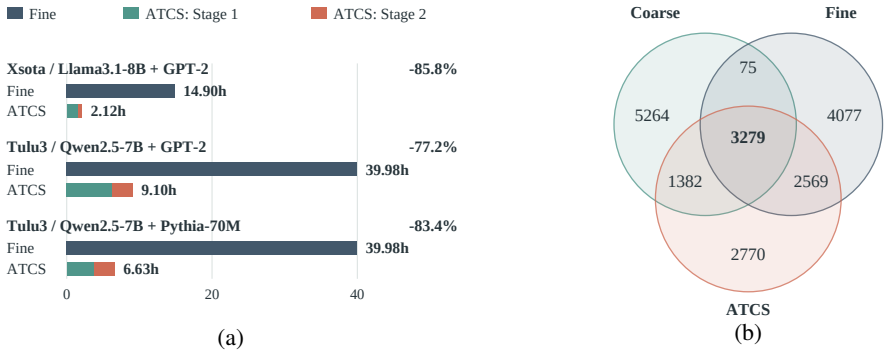


Figure 3: (a) IFD selection time on one RTX 4090. (b) Overlap of 10K Xsota coresets under NLL.

target model's selection preference. Stage 2 raises the overlap with Fine to 5,848 samples, bringing ATCS substantially closer to exhaustive target-model selection.

4.2 Ablation Studies

Cross-utility selection and redundancy. We evaluate all six assignments of different utilities to proxy filtering and target-model reranking. In Table 2(a), rows denote Stage 1 and columns Stage 2, under the Xsota/Llama3.1-8B/GPT-2/10K setting. Mixed settings generally outperform the homogeneous setting using the same utility, indicating that cross-utility selection can yield additional gains in some settings. Besides, we also observe unusually poor Fine performance under IFD on Tulu3/Qwen2.5-7B/GPT-2 in Table 1 and Appendix Table 4 (the Tulu3/Qwen2.5-7B/Pythia-70M setting). We hypothesize that redundant selections may contribute to this result and therefore conduct a within-coreset redundancy analysis. Table 2(b) shows that Fine yields more redundant coresets than ATCS under each homogeneous utility (Mean NN is the average nearest-neighbor cosine similarity, while Redundancy@0.8 is the percentage of samples whose nearest-neighbor similarity is at least 0.8, more details are provided in Appendix B). This highlights the risk that exact maximization of a single utility can concentrate the selected coreset. The two cross-utility configurations

Table 2: Cross-utility performance and coreset redundancy.

(a) downstream performance

Stage 1	Stage 2	Avg.	Gain
NLL	NLL	59.72	–
IFD	IFD	59.88	–
UFS	UFS	58.86	–
IFD	NLL	**60.97**	+1.25
IFD	UFS	59.14	+0.28
NLL	IFD	59.90	+0.02
NLL	UFS	59.45	+0.59
UFS	IFD	60.92	+1.04
UFS	NLL	59.68	-0.04

(b) Tulu3: coreset redundancy

Utility	Method	Mean NN	Red.@0.8
NLL	Fine	0.618	14.7%
	ATCS	0.587	7.7%
IFD	Fine	0.643	21.7%
	ATCS	0.579	6.5%
UFS	Fine	0.609	11.9%
	ATCS	0.546	5.3%
NLL–IFD	ATCS	**0.512**	**4.9%**
UFS–IFD	ATCS	0.524	5.0%

Table 3: Key sentences vs. length-matched truncation on Xsota. ΔAvg. is relative to key sentences.

Utility	Scoring input	ARC-C	BBH	IFEval	MMLU	Avg.	ΔAvg.
NLL	Key sentences	**76.95**	**62.76**	**40.13**	**59.04**	**59.72**	–
	Left truncation	76.15	61.97	36.72	55.71	57.64	-2.08
	Right truncation	76.33	62.55	38.22	57.45	58.64	-1.08
IFD	Key sentences	**79.32**	**62.79**	**40.53**	56.89	**59.88**	–
	Left truncation	77.43	61.14	35.86	54.32	57.19	-2.69
	Right truncation	76.89	62.03	37.77	**57.02**	58.43	-1.45
UFS	Key sentences	77.97	**63.36**	**38.30**	**55.79**	**58.86**	–
	Left truncation	77.22	61.23	34.64	54.96	57.01	-1.85
	Right truncation	**78.34**	61.54	36.44	55.10	57.86	-1.00

evaluated for redundancy achieve lower Redundancy@0.8 (4.9% and 5.0%) than all three homogeneous ATCS settings (5.3–7.7%), suggesting that mixing utilities can partly mitigate this risk. These findings motivate future selection strategies that consider not only the value of individual samples to the target model, but also the collective value of the selected coreset.

Key sentences versus positional truncation. To test whether attention-guided localization improves selection beyond simply shortening the scoring input, we compare it with left and right truncation under the same maximum scoring-length budget, keeping all other selection components fixed. Shortening is applied only during selection-time scoring, and all methods fine-tune on the selected full examples. Table 3 shows that key sentences outperform both truncation baselines under NLL, IFD, and UFS. This suggests that retaining response-relevant content improves utility estimation beyond merely shortening the input. Such content can be scattered throughout an instruction, so a fixed prefix or suffix may omit information that key-sentence localization preserves.

5 Conclusion

This work revisits instruction data selection from an efficiency perspective. Our findings show that the cost of target-model-aware selection can be substantially reduced by allocating target-model computation selectively across both samples and tokens. A lightweight proxy is sufficient to identify promising candidates and localize response-relevant instruction content, while limited target-model reranking recovers much of the benefit of exhaustive scoring. This makes target-model-aware selection practical without replacing its underlying utility function. Our experiments further reveal that exhaustive maximization of a single utility is not necessarily optimal, as it may concentrate selection on a narrow and redundant subset. The effectiveness of cross-utility configurations suggests that filtering and reranking can provide complementary signals. These findings motivate future selection methods that jointly account for computational cost, sample value to the target model, and diversity within the resulting coreset.

AI USE STATEMENT

We used large language models for language polishing, grammar checking, and assistance with code debugging. All research ideas, experimental design, and experiments were developed and conducted by the authors. The authors reviewed all AI-assisted outputs and take full responsibility for the manuscript and code.

REFERENCES

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *KDD*, pp. 1119–1127, 2018.

Stella Biderman et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430, 2023.

Chengliang Chai, Jiayi Wang, Nan Tang, Ye Yuan, Jiabin Liu, Yuhao Deng, and Guoren Wang. Efficient coreset selection with cluster-based methods. In *KDD*, pp. 167–178. ACM, 2023.

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

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

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

Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

HuggingFaceH4. UltraChat 200k dataset. `https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k`, 2023.

Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *Proc. VLDB Endow.*, 9(12):948–959, 2016.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*, 2024a.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7602–7635, 2024b.

Zhuang Li, Yuncheng Hua, Thuy-Trang Vu, Haolan Zhan, Lizhen Qu, and Gholamreza Haffari. SCAR: data selection via style consistency-aware response ranking for efficient instruction-tuning of large language models. In *ACL (1)*, pp. 12756–12790. Association for Computational Linguistics, 2025.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. Selectit: Selective instruction tuning for large language models via uncertainty-aware self-reflection. *CoRR*, abs/2402.16705, 2024a.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=BTKAeLqLMw.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *ICLR*. OpenReview.net, 2024.

OpenCompass Contributors. OpenCompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. 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. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In *OpenAI Blog*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: rapid training data creation with weak supervision. *The VLDB Journal*, 29(2):709, 2019.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 12, pp. 223, 2018.

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

An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jipeng Zhang, Yaxuan Qin, Renjie Pi, Weizhong Zhang, Rui Pan, and Tong Zhang. Tagcos: Task-agnostic gradient clustered coreset selection for instruction tuning data. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4671–4686, 2025.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Pretraining data detection for large language models: A divergence-based calibration method. *arXiv preprint arXiv:2409.14781*, 2024.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 400–410, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

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. LIMA: less is more for alignment. In *NeurIPS*, 2023a.

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

A Datasets, Models, and Evaluation Benchmarks

Instruction-tuning datasets.

- **Xsota** (Lu et al., 2024): We use the `deita_sota_pool` version, which contains approximately 300K instruction–response pairs aggregated from multiple high-quality instruction datasets. It covers diverse tasks including question answering, reasoning, coding, and open-ended dialogue.
- **Tulu3 SFT mixture** (Lambert et al., 2024): This dataset contains approximately 939K instruction–response pairs collected from heterogeneous sources. It includes general instruction following, knowledge-intensive question answering, mathematical and logical reasoning, coding, and multilingual tasks.
- **UltraChat-200K** (HuggingFaceH4, 2023): This dataset is a curated subset of approximately 200K multi-turn instructional conversations. It contains user–assistant interactions spanning information seeking, content generation, reasoning, and open-ended dialogue.

Target models.

- **Llama3.1-8B** (Dubey et al., 2024): A decoder-only language model from the Llama 3.1 family with 8B parameters. It supports long-context and multilingual generation and provides strong general-purpose instruction-following capabilities.
- **Qwen2.5-7B** (Yang et al., 2024): A 7B-parameter decoder-only language model from the Qwen2.5 family, designed to support multilingual understanding, reasoning, coding, and instruction following.
- **Qwen2.5-14B** (Yang et al., 2024): The 14B-parameter variant of Qwen2.5. It shares the model family and training design of Qwen2.5-7B while providing greater model capacity.

Proxy models.

- **GPT-2** (Radford et al., 2019): A 124M-parameter autoregressive Transformer pretrained with a standard next-token prediction objective.
- **Pythia-70M** (Biderman et al., 2023): A 70M-parameter decoder-only language model from the Pythia suite, which provides openly released training data, configurations, and intermediate checkpoints for studying language-model behavior.

Evaluation benchmarks.

- **ARC-C** (Clark et al., 2018): The challenge subset of the AI2 Reasoning Challenge contains 1,172 multiple-choice questions drawn from grade-school science exams. These questions are selected because retrieval- and word-co-occurrence baselines fail to answer them, making the subset more reasoning-intensive than ARC-Easy. We report zero-shot answer accuracy.
- **BBH** (Suzgun et al., 2023): BIG-Bench Hard comprises 23 challenging tasks involving logical, algorithmic, linguistic, and multistep reasoning. We report the average zero-shot score across these tasks.
- **IFEval** (Zhou et al., 2023b): Evaluates whether model outputs satisfy verifiable constraints, such as required formats, keywords, lengths, and structural rules. The benchmark contains 541 prompts covering 25 types of verifiable instructions. Compliance is determined using its rule-based evaluators in the zero-shot setting.
- **MMLU** (Hendrycks et al., 2020): Contains multiple-choice questions from 57 subjects spanning humanities, social sciences, STEM, and professional knowledge. Its subjects range from elementary mathematics and world history to computer science and law, covering both factual knowledge and problem-solving ability. We report zero-shot accuracy averaged across subjects.
- **GSM8K** (Cobbe et al., 2021): Evaluates multistep mathematical reasoning on grade-school word problems. We use eight in-context demonstrations for each test query and report exact-match accuracy of the final answer.
- **AlpacaEval** (Dubois et al., 2024): Evaluates open-ended instruction following. GPT-5.4 performs pairwise comparisons between outputs from the evaluated model and the specified reference model, and the percentage of preferred outputs is reported as the win rate.

B EXPERIMENTAL SETUP

Baselines.

- **Random**: Uniformly samples the required number of examples. We report the mean over three independently sampled coresets.
- **Coarse**: Selects the coreset directly according to utility scores computed by the proxy model, without target-model reranking.
- **Fine**: Exhaustively computes utility scores for the full candidate pool using the target model and selects the highest-ranked examples.
- **Superfiltering** (Li et al., 2024a): We follow its proxy-only selection procedure and apply utility-specific outlier filtering. For IFD, we discard samples with scores above one, as in the original method. For NLL and UFS, we remove the top 5% highest-scoring samples as utility-specific outliers. Following the official implementation, sequences are right-truncated to at most 1,024 tokens.
- **FastCore** (Chai et al., 2023): Approximates gradient-based coreset selection by partitioning samples into clusters and selecting representatives that cover the resulting feature space. It uses sample-to-cluster distances and product quantization to reduce the distance computations required during selection. In our implementation, token embeddings are used as sample features.

Selection utilities. For a sample (x_i, y_i), let $\mathcal{M}$ denote the scoring model and N_y the number of response tokens.

- **NLL**: Measures the average negative log-likelihood of the response conditioned on the instruction:

$$\mathrm{NLL}_{\mathcal{M}}(x_i, y_i) = -\frac{1}{N_y} \sum_{t=1}^{N_y} \log p_{\mathcal{M}}(y_{i,t} \mid x_i, y_{i,<t}) \,. \tag{7}$$

  A higher NLL indicates that the response is less predictable under $\mathcal{M}$.
- **IFD** (Li et al., 2024b): Compares the response perplexity with and without the instruction:

$$\mathrm{IFD}_{\mathcal{M}}(x_i, y_i) = \frac{\mathrm{PPL}_{\mathcal{M}}(y_i \mid x_i)}{\mathrm{PPL}_{\mathcal{M}}(y_i)} = \exp(\mathrm{NLL}_{\mathcal{M}}(y_i \mid x_i) - \mathrm{NLL}_{\mathcal{M}}(y_i)) \,. \tag{8}$$

  A higher IFD indicates that the instruction provides less assistance in predicting the response.
- **UFS** (Zhang et al., 2024): Following DC-PDD, let $z_i = [x_i; y_i]$, $q_{i,t} = p_{\mathcal{M}}(z_{i,t} \mid z_{i,<t})$, and $f_{\mathcal{R}}(z_{i,t})$ denote the empirical frequency of token $z_{i,t}$ in a reference corpus $\mathcal{R}$. We compute

$$c_{i,t} = \min\left\{ q_{i,t} \log \frac{1}{f_{\mathcal{R}}(z_{i,t})},\, a \right\}, \tag{9}$$

  where $a = 0.01$ limits the contribution of an individual token. Let $\mathcal{F}_i = \{t : z_{i,t} \notin z_{i,<t}\}$ contain the first-occurrence indices, as repeated occurrences of the same token are excluded. We define

$$\mathrm{UFS}_{\mathcal{M}}(z_i) = -\frac{1}{|\mathcal{F}_i|} \sum_{t \in \mathcal{F}_i} c_{i,t}, \tag{10}$$

  where a higher value indicates greater unfamiliarity to $\mathcal{M}$. We estimate $f_{\mathcal{R}}$ with Laplace smoothing from the first 15 shards of the C4 training split (Raffel et al., 2020).

Fine-tuning configuration. Unless otherwise specified, each method selects a 10K coreset. For ATCS, the coarse stage retains the top $\alpha = 0.1$ candidates, key sentences are selected using a cumulative attention threshold of $\tau = 0.7$, and utility estimation uses at most the first 50 response tokens. Selected examples are restored to their complete original form before SFT.

We fine-tune all target models for five epochs using LoRA implemented with LLaMA-Factory (Zheng et al., 2024). We use a learning rate of 5×10^{-5}, a cosine learning-rate scheduler with 60 warmup steps, and FP16 precision. All selection methods use the same fine-tuning configuration.

Evaluation configuration. ARC-C, BBH, IFEval, and MMLU are evaluated through OpenCompass (OpenCompass Contributors, 2023) using zero-shot prompting. GSM8K is evaluated with eight-shot prompting and exact-match accuracy. For AlpacaEval, GPT-5.4 performs pairwise comparisons between outputs from each evaluated model and those from the Random baseline, and we report the resulting win rate. The average in Table 1 is the unweighted mean of all six benchmark scores, whereas Table 4 reports the mean of ARC-C, BBH, IFEval, and MMLU.

Analysis-specific settings.

- For the upper part of Figure 2a, we sample 10K examples from each dataset and apply the Stage 1 length-control procedure. Each model independently computes NLL. For $q \in \{0.2, 0.3, 0.4, 0.5\}$, we report $\mathrm{Overlap@}q = |B_q^{(m_1)} \cap B_q^{(m_2)}|/|B_q^{(m_1)}|$, whose random expectation is q.
- For the lower part of Figure 2a, Jaccard@k measures the intersection-over-union of the two top-k sentence sets, while Recall@k measures the fraction of the second model's top-k sentences recovered by the first model, where $k \in \{3, 5\}$.
- For Figure 2b, we use GPT-2 on Xsota and vary the cumulative attention threshold τ from 0.5 to 1.0. We report the response NLL and the retained instruction-token ratio at each threshold.
- For Figure 3a, we measure wall-clock selection time on a single RTX 4090. ATCS includes proxy filtering and target-model reranking, whereas Fine includes exhaustive target-model scoring. SFT and downstream evaluation are excluded.
- For Table 2(b), we analyze 10K Tulu3 coresets using Qwen2.5-7B as the target model and GPT-2 as the proxy. Instructions are encoded using `all-MiniLM-L6-v2` with mean pooling and ℓ_2 normalization. Mean NN is the mean nearest-neighbor cosine similarity, and Redundancy@0.8 is the percentage of instructions whose nearest-neighbor similarity is at least 0.8.

C Additional Experimental Results

Table 4: Downstream performance across additional settings.

Utility	Random	Coarse	Fine	FastCore	Superfiltering	ATCS
Xsota / Llama3.1-8B / GPT-2 / 10K						
NLL	56.18	58.80	58.82	58.62	58.62	**59.72**
IFD	56.18	58.45	59.35	58.25	58.34	**59.88**
Tulu3 / Qwen2.5-7B / GPT-2 / 10K						
NLL	47.95	49.02	**51.59**	48.86	49.32	51.49
UFS	47.95	51.69	51.41	52.18	51.51	**52.83**
Tulu3 / Qwen2.5-7B / Pythia-70M / 10K						
NLL	47.95	44.17	51.59	45.48	45.19	**52.25**
IFD	47.95	48.98	39.26	**49.32**	49.11	49.25
UFS	47.95	50.60	51.18	50.50	50.60	**52.80**
Xsota / Llama3.1-8B / GPT-2 / 20K						
NLL	57.87	60.36	61.00	60.53	60.40	**61.21**
IFD	57.87	58.85	59.29	59.03	59.00	**59.60**
UFS	57.87	58.51	59.06	58.19	58.16	**59.42**
Tulu3 / Qwen2.5-14B / GPT-2 / 10K						
NLL	47.30	49.01	57.59	50.87	49.09	**57.60**
IFD	47.30	48.81	**56.05**	51.10	49.17	54.66
UFS	47.30	48.73	**55.38**	50.20	49.83	55.06

Results across utilities and experimental settings. Table 4 presents additional model, proxy, utility, and coreset-size settings for the downstream performance experiments reported in Section 4.1. The reported results are averages over ARC-C, BBH, IFEval, and MMLU. All settings follow the same training and evaluation protocol described above. ATCS demonstrates consistently strong performance across the evaluated utilities, proxy models, coreset budgets, and target-model scales.

Table 5: Hyperparameter sensitivity. † denotes the default setting.

Parameter	Value	ARC-C	BBH	IFEval	MMLU	Avg.
α	0.05	76.95	63.16	44.38	54.69	**59.80**
	0.10†	76.95	62.76	40.13	59.04	59.72
	0.20	78.98	62.75	38.17	54.58	58.62
	0.30	78.98	63.37	38.68	46.76	56.95
τ	0.50	76.95	63.16	39.93	57.23	59.32
	0.60	78.64	62.56	39.44	59.17	**59.95**
	0.70†	76.95	62.76	40.13	59.04	59.72
	0.80	79.66	61.55	41.24	55.52	59.49
T_y	50†	76.95	62.76	40.13	59.04	59.72
	100	78.32	62.35	40.72	58.47	59.97
	150	77.95	62.95	40.40	57.05	59.59
	200	77.95	65.03	40.16	56.81	**59.99**

Hyperparameter sensitivity. We vary one hyperparameter at a time under the Xsota/GPT-2/Llama3.1-8B/NLL/10K setting, as shown in Table 5. Here, α is the fraction of candidates retained for target-model reranking, τ is the cumulative attention threshold used to select key instruction sentences, and T_y is the maximum number of response tokens used for utility scoring.

The results at $\alpha = 0.05$ and 0.1 are similar, whereas increasing α to 0.2 and 0.3 reduces the average score to 58.62 and 56.95 while requiring the target model to score more candidates. Performance varies by only 0.63 points across the tested τ values and by 0.40 points across response lengths. Using only the first 50 response tokens remains competitive with substantially longer inputs. These results show that ATCS is stable with respect to τ and response length, while a small α provides a favorable balance between downstream performance and target-model scoring cost.