

RETHINKING LLM EVALUATION: CAN WE EVALUATE LLMs WITH $200\times$ LESS DATA?

Shaobo Wang ^{†*1,2,3} Cong Wang ^{*1} Wenjie Fu ^{*1,4} Yue Min ^{1,5} Mingquan Feng ² Isabel Guan ⁵ Xuming Hu ^{5,6}
 Conghui He ⁷ Cunxiang Wang ⁸ Kexin Yang ³ Xingzhang Ren ³ Fei Huang ³ Dayiheng Liu ³ Linfeng Zhang ^{✉1,2}
¹ EPIC Lab, SJTU ² SJTU ³ Alibaba Group ⁴ FDU ⁵ HKUST ⁶ HKUST (GZ)
⁷ Shanghai AI Lab ⁸ ZhipuAI ^{*} Equal contribution [✉] Corresponding author [†] Project Head

ABSTRACT

As the demand for comprehensive evaluations of diverse model capabilities steadily increases, benchmark suites have correspondingly grown significantly in scale. Despite notable advances in redundancy reduction and subset-level performance prediction, a systematic framework that effectively integrates these methods to ensure both prediction accuracy and ranking consistency is still largely elusive. In this paper, we first perform a sample-level analysis of benchmark redundancy and identify several highly similar samples that can be eliminated. Besides, we frame benchmark compression as an optimization problem with the aim of score reconstruction. Building on these, we then propose EssenceBench, a coarse-to-fine framework utilizing an iterative Genetic Algorithm (GA), which takes the advantages of fitness-based subset search and attribution-based sample search. Compared to previous methods, our approach yields superior compression results with lower reconstruction error and markedly higher efficiency. In particular, on the HellaSwag benchmark (10K samples), our method preserves the ranking of all models shifting within 5% using $25\times$ fewer samples, and achieves 95% ranking preservation shifting within 5% using only $200\times$ fewer samples.

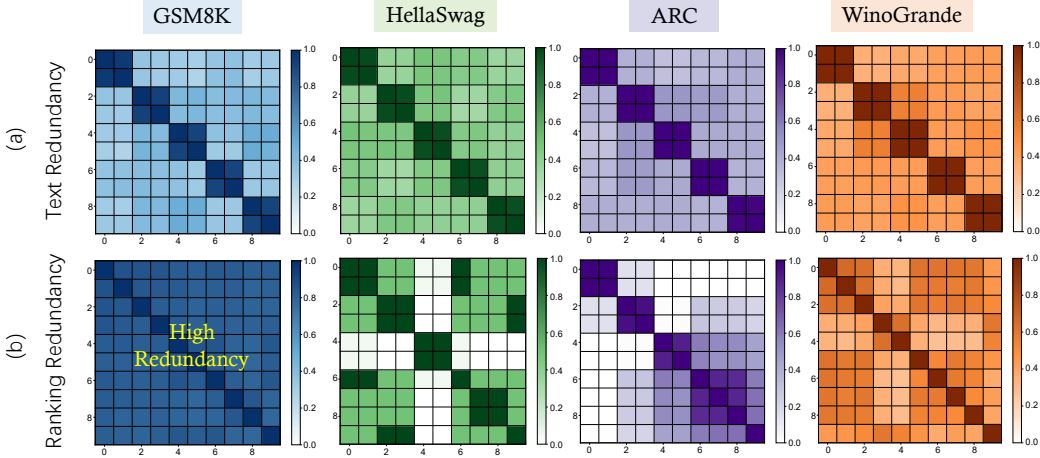


Figure 1: Prevalent redundancy across widely used benchmark datasets. Based on 10 randomly sampled instances per dataset, panel (a) depicts the *text embedding similarity* (Definition 3), reflecting semantic overlap among instances, and panel (b) presents the *ranking embedding similarity* (Definition 4), measured through consistency of model performance rankings across sampled subsets.

1 INTRODUCTION

In recent years, large language models (LLMs) have advanced rapidly with the release of models such as GPT-4 (Achiam et al., 2023). This swift progress has led to a shift in increasingly sophisticated LLM benchmark design, moving from traditional natural language processing (NLP) tasks to more comprehensive, multidimensional evaluation suites. Prominent benchmarks have been released

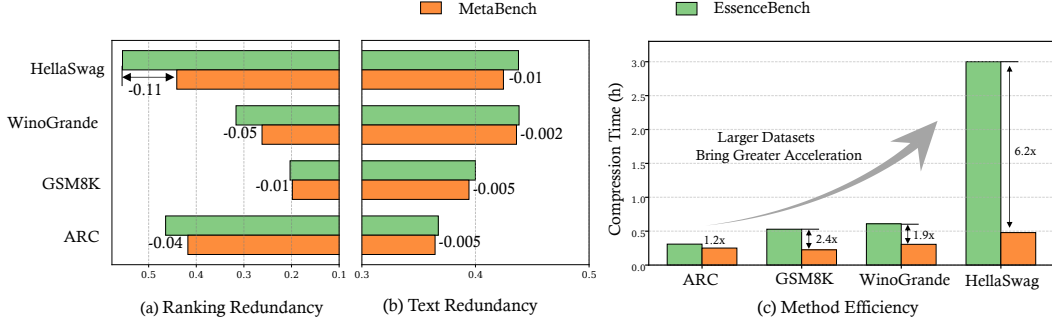


Figure 2: Comparison of existing benchmark compression approaches and our EssenceBench. (a) ranking and (b) text redundancy comparison and (c) compression time comparison.

to evaluate LLMs in areas such as multilingual understanding (Zhao et al.), long-context reasoning (Kuratov et al., 2024), instruction following (Yin et al., 2023), mathematical reasoning (Shao et al., 2024), code comprehension and generation (Nam et al., 2024), multidisciplinary knowledge acquisition (Zhang et al., 2025), and tool integration (Chen et al., 2024b). However, as the scope and granularity of evaluation expand, so does the scale and computing cost of the benchmarking process. For instance, *OpenCompass* (Contributors, 2023) integrates over 60 subtasks across more than 25 capability dimensions. Evaluating *Qwen2.5-7B-Instruct*¹ across all these tasks often takes about **1k GPU hours**, consuming millions to tens of millions of tokens. Consequently, the question of **how to efficiently reduce the sample size of benchmark datasets while preserving the reliability of evaluation** has become a critical challenge in the current phase of LLM.

We first revisit the foundations of LLM benchmark evaluation, focusing on a critical yet under-examined phenomenon: *sample redundancy* in the *Open LLM Leaderboard*² (Fourrier et al., 2024). Our analysis quantifies redundancy through two complementary dimensions: (i) **Text-level redundancy**: Defined as lexical and semantic overlap between evaluation instances (Definition 3) and (ii) **Ranking-level redundancy**: Measured through consistency of model performance rankings across sampled subsets (Definition 4). As shown in Figure 1, we systematically evaluate redundancy across benchmark datasets by randomly sampling 10 instances per dataset. Our analysis reveals significant redundancy patterns that persist across diverse benchmark configurations and model architectures, manifesting in both textual content and performance ranking dimensions.

Recent studies have explored benchmark compression. Methods such as LLM-based annotation (Li et al., 2024), active sample querying (Kossen et al., 2021; Polo et al., 2024), and psychological approaches (Polo et al., 2024; Kipnis et al., 2025) have been employed to reduce benchmark dataset size while maintaining evaluation quality. Notably, *MetaBench* (Kipnis et al., 2025) uses Generalized Additive Models (GAM) (Hastie, 2017) to model the relationship between subset scores and full set performance, and it employs root mean square error (RMSE) as an index to guide sampling policies. Despite their contributions, previous works have two significant limitations:

1. **Neglect of Sample Interactions**: Conventional approaches treat test samples as independent entities, ignoring semantic relationships. Specifically, when two samples exhibit high similarity, their evaluation outcomes often correlate strongly, suggesting a redundancy that warrants systematic elimination. Developing robust metrics and principled approaches for identifying such redundancies remains an open challenge.
2. **Inefficient Search Mechanisms**: Existing compression techniques rely on statistical or heuristic methods (e.g., GAM, LLM scoring, active querying) that suffer from high computational overhead or suboptimal convergence.

To tackle the challenge of redundant and costly LLM benchmark evaluations, as shown in Figure 2, we introduce *EssenceBench*, a *coarse-to-fine, iterative compression framework* that preserves evaluation fidelity while reducing dataset size. As illustrated in Figure 3, (i) we first extract each benchmark’s score matrix from the *Open LLM Leaderboard*. To weight the interactions between samples and eliminate the first limitation, we quantify text-level and ranking-level redundancies via embedding

¹<https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>

²<https://huggingface.co/datasets/open-llm-leaderboard-old/results>

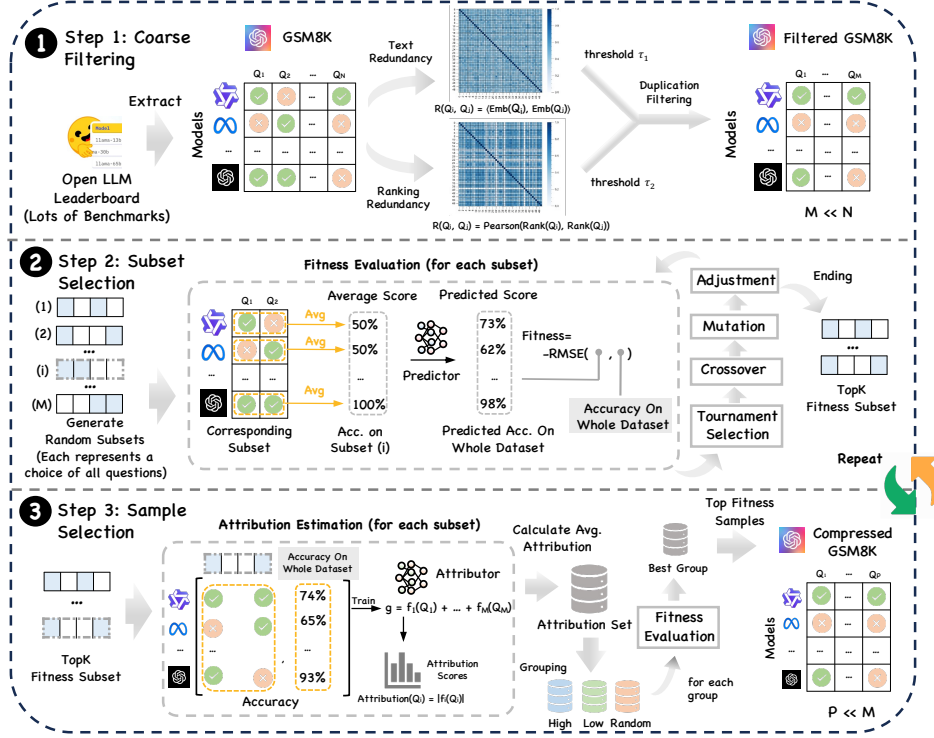


Figure 3: The pipeline of EssenceBench. (I) **Coarse Filtering**. By extracting the binary score matrix for each benchmark and computing both text-level and ranking-level redundancies, samples that exceed thresholds are removed. (II) **Subset Selection**. Genetic Algorithm (GA) is applied beginning with generating random subsets. With fitness evaluated by the error of predicted accuracy, subsets are optimized via fitness-based tournament selection, crossover, mutation, and adjustment. (III) **Sample Selection**. Attribution of each sample is estimated from the top-performing subsets by utilizing weights when training a model. According to that, samples are divided into groups. GA is then reapplied within each group to identify the most representative and informative subset.

similarities and ranking correlations, then eliminate duplicate samples. (ii) On this filtered set, we launch an iterative Genetic Algorithm (GA) to identify compact yet representative subsets: in each GA run, candidate subsets are evaluated by a predictor model to minimize the prediction error against full-dataset accuracy, guiding the search toward subsets that faithfully reconstruct overall performance. (iii) Finally, to resolve the inefficiency of search mechanisms, we refine our selection through sample-level attribution, using the weights learned during model training to partition instances into high, low, and random attribution groups; we reapply GA within each group and choose the top-performing samples from the best group as our compressed benchmark for that round. This further searching step introduces sample-level diversity thus alleviating suboptimal convergence. By combining redundancy-aware filtering with iterative GA optimization, EssenceBench delivers benchmarks that are both lean and reliable, enabling rapid, cost-effective evaluation of cutting-edge LLMs. Our contributions are as follows:

- We systematically analyze redundancy problems in LLM benchmarks and observe that all benchmarks in Open LLM Leaderboard share a sample redundancy phenomenon, which causes evaluation inefficiency.
- We frame benchmark compression as an optimization problem and tackle with it effectively by combining redundancy-based coarse filtering and iterative Genetic Algorithm. The proposed framework, EssenceBench, efficiently addresses this problem while ensuring scalability.
- Experimental results demonstrate that EssenceBench achieves significant reductions while efficiently maintaining rankings. Notably, on HellaSwag (10K samples), our method preserves the ranking of all models shifting within 5% using $25\times$ fewer samples.

2 RELATED WORKS

Large Language Model Evaluation. Benchmarks are indispensable for measuring LLM capabilities and catalyzing research. Standard NLU and reading comprehension tasks include GLUE (Wang et al.) and SQuAD (Rajpurkar et al.); mathematical reasoning is tested by GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al.) and Mathqa (Amini et al., 2019); coding ability via HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021); instruction following by IFEval (Zhou et al., 2023b); and multiple-choice and commonsense reasoning by MMLU (Hendrycks et al., 2020), ARC-Challenge (Clark et al., 2018) and HellaSwag (Zellers et al., 2019). Platforms such as Open LLM Leaderboard (Fourrier et al., 2024) and OpenCompass (Contributors, 2023) provide unified pipelines and live leaderboards, but rarely consider the cost and efficiency.

LLM Benchmark Compression. As benchmarks scale up, compact evaluation suites are critical (Li et al., 2024). TinyBenchmark (Polo et al., 2024) combines statistical selection with Item Response Theory (IRT) to pick 100 representative items; MetaBench (Kipnis et al., 2025) uses IRT and Fisher information over 5000 LLM outputs to select discriminative examples, achieving an average RMSE of 1.5%. However, metaheuristic search techniques such as Genetic Algorithms have not been explored, and the redundancy phenomenon highlighted in (Zhang et al., 2025; Li et al., 2025) remains insufficiently addressed. For a broader survey of recent advances in LLM data selection and compression, we refer readers to Appendix A.

3 METHODOLOGY

3.1 PRELIMINARIES: BENCHMARK COMPRESSION

The benchmark compression problem can be framed as selecting a k -element subset (a coreset) from the universal set (the benchmark). If the goal is to reconstruct the score of the benchmark, finding the subset requires searching over an exponentially large, discrete space of candidate subsets, which makes it a classic NP-hard combinatorial optimization task. Formally, it can be defined as:

Definition 1 (Benchmark Compression). *Let $\mathcal{D} = \{x_1, \dots, x_N\}$ be a benchmark dataset, and let $g : 2^{\mathcal{D}} \rightarrow \mathbb{R}$ be an aggregate scoring function that assigns a performance score to any subset of $\mathcal{D}$. Given a budget $k < N$, the compression problem seeks a k -element subset $\tilde{\mathcal{D}} \subseteq \mathcal{D}$ that best reconstructs the full-dataset score:*

$$\tilde{\mathcal{D}}^* = \arg \min_{\tilde{\mathcal{D}} \subseteq \mathcal{D}, |\tilde{\mathcal{D}}|=k} \mathcal{L}(g(\mathcal{D}), g(\tilde{\mathcal{D}})), \quad (1)$$

where $\mathcal{L}(\cdot, \cdot)$ represents a suitable error measure.

However, Definition 1 relies on a function to impractically traverse all subsets of the whole dataset. To effectively overcome this limitation, we leverage the fact that public leaderboards (Fourrier et al., 2024) report a binary correctness (right or wrong) for each model on each benchmark sample. By systematically organizing these outcomes into a *score matrix*, we simplify the scoring function into column selection and accuracy computation. Let $\mathcal{D}$ be the benchmark and N_{LLM} denote the number of LLMs tested on it. The *score matrix* is denoted as: $\mathbf{S} \in \{0, 1\}^{N_{\text{LLM}} \times N}$, where $\mathbf{S}_{i,j}$ explicitly denotes the score of LLM _{i} on sample x_j , 1 for right and 0 for wrong. Let $\mathbf{y} \in \mathbb{R}^{N_{\text{LLM}}}$ denote the accuracy of LLMs on $\mathcal{D}$, where $y_i = \frac{1}{N} \sum_{j=1}^N \mathbf{S}_{i,j}$ is the accuracy of LLM _{i} . The concrete formulation of benchmark compression is then defined in Definition 2.

Definition 2 (Concrete Formulation of Benchmark Compression). *Let a binary mask represents the selection of a k -element subset: $\mathbf{m} \in \{0, 1\}^N$ s.t. $\sum_{j=1}^N m_j = k$. By indexing the mask, the matching columns of $\mathbf{S}$ can be got, which is denoted as $\mathbf{S}_{\mathbf{m}}$. The aggregate scoring function g in Definition 1 can be concretized as: $g(\tilde{\mathcal{D}}) = g(\mathbf{S}_{\mathbf{m}})$. Therefore, the optimization problem is as:*

$$\min_{\mathbf{m} \in \{0, 1\}^N} \mathcal{L}(\mathbf{y}, g(\mathbf{S}_{\mathbf{m}})) \quad \text{s.t.} \quad \sum_{j=1}^N m_j = k. \quad (2)$$

3.2 A CLOSER LOOK AT THE SAMPLE REDUNDANCY PHENOMENON IN LLM BENCHMARKS

As illustrated in Figure 1, many LLM benchmarks exhibit high overlap both in the text of their prompts and in the models’ performance rankings in our quantification of sample redundancy, *i.e.* text redundancy (Definition 3) and ranking redundancy (Definition 4). Intuitively, when two examples share similar wording or contain nearly identical behavior across some models, retaining both adds little new information while doubling evaluation costs.

Definition 3 (Sample Redundancy from Text Perspective). *Let $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ be a benchmark dataset where each x_i denotes the textual input. Let $\text{Emb} : x \mapsto \mathbb{R}^{d_{\text{emb}}}$ denote an embedding mapping of the input. The redundancy of a specific sample pair is defined as:*

$$\mathcal{R}_{\text{text}}(i, j) = \langle \text{Emb}(x_i), \text{Emb}(x_j) \rangle. \quad (3)$$

The redundancy of a sample is then defined as:

$$\mathcal{R}_{\text{text}}(i) = \frac{\sum_{j \neq i} \langle \text{Emb}(x_i), \text{Emb}(x_j) \rangle}{N - 1}. \quad (4)$$

The overall redundancy of the benchmark is defined as the average redundancy across all samples:

$$\mathcal{R}_{\text{text}}(\mathcal{D}) = \frac{\sum_{i=1}^N \mathcal{R}_{\text{text}}(i)}{N}. \quad (5)$$

Definition 4 (Sample Redundancy from Ranking Perspective). *Let $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ be a benchmark dataset. Suppose we are given a set of responses from several LLMs, and for each sample x_i , we define a ranking score $r_i \in \mathbb{R}$ indicating the model’s confidence or correctness on that sample. For example, r_i could be a binary indicator (e.g., correct/incorrect) or a continuous score (e.g., answer log-likelihood). We define the redundancy between two samples x_i and x_j in terms of their ranking correlation as:*

$$\mathcal{R}_{\text{ranking}}(i, j) = \rho(r_i, r_j) \quad (6)$$

where $\rho(\cdot, \cdot)$ denotes the correlation coefficient, which can be Pearson, Spearman or R^2 . The one-sample and overall redundancy are defined the same way as:

$$\mathcal{R}_{\text{ranking}}(i) = \frac{\sum_{j \neq i} |\rho(r_i, r_j)|}{N - 1}, \quad \mathcal{R}_{\text{ranking}}(\mathcal{D}) = \frac{\sum_{i=1}^N |\rho(r_i, r_j)|}{N}. \quad (7)$$

These two definitions are intuitively derived from both human and LLM perspectives. Specifically, textual redundancy quantifies semantic similarity, while ranking redundancy assesses behavioral similarity. When combined, they reveal complementary overlaps that each alone would overlook, facilitating more systematic redundancy elimination and benchmark pruning.

3.3 ESSENCEBENCH

Step1: Coarse Filtering. On the basis of *text redundancy* (Definition 3) and *ranking redundancy* (Definition 4). The benchmark $\mathcal{D}$ with size N can be filtered through the threshold τ_{text} and τ_{ranking} to size M as shown in Figure 3. We examine the dataset in its original order and decide for each sample x_i whether to keep or discard it. For each sample x_j , if either $\mathcal{R}_{\text{text}}(j, i) > \tau_{\text{text}}$ or $\mathcal{R}_{\text{ranking}}(j, i) > \tau_{\text{ranking}}$, then x_i is discarded; otherwise, it is retained. This ensures that among any highly redundant pair, the sample encountered first is always kept. Let ϵ_i be the flag indicating whether x_i should be discarded, formalized as:

$$\epsilon_i = \prod_{j=1}^{i-1} \mathbf{1}(\mathcal{R}_{\text{text}}(j, i) \leq \tau_{\text{text}} \wedge \mathcal{R}_{\text{ranking}}(j, i) \leq \tau_{\text{ranking}}), \quad (8)$$

where $\mathbf{1}(\cdot)$ denotes the indicator function. Therefore, the filtered benchmark is $\mathcal{D}_{\text{filtered}} = \{x_i \mid \epsilon_i = 1\}$ with size M . In this process, we strip away the most redundant examples to yield a compact yet representative filtered set, lightening our evaluation load and making compression faster.

Step2: Fitness-based Subset Selection. To shrink the filtered benchmark while preserving its ability to approximate the benchmark score, we employ an iterative genetic algorithm (Lambora et al., 2019; Mirjalili & Mirjalili, 2019; Mathew, 2012; Mitchell, 1998) as a heuristic search over the space of possible subsets. As Figure 3 shows, starting from a randomly initialized *population* of k -element masks, we repeatedly evaluate each mask’s *fitness*, select parents via *tournament selection*, generate offspring through *crossover* and *mutation*, and then *adjust* each child to enforce the k -element

Table 1: Prediction Error ($\downarrow$) of selected subsets with different sizes.

Dataset	Method	Coreset Size									
		50	100	150	200	250	300	350	400	450	500
GSM8K	Random	3.6894	2.8531	2.2934	1.9454	1.6267	1.4013	1.3432	1.2279	1.1487	1.0400
	PPL	4.1941	2.6987	2.3007	1.9218	1.719	1.5153	1.3588	1.3002	1.1988	1.1301
	GraNd	3.8516	2.8929	2.3725	2.0956	1.8256	1.5994	1.4442	1.2719	1.1568	1.0692
	MetaBench	3.5283	2.4335	2.0673	1.7597	1.5529	1.3873	1.2631	1.1301	1.0333	0.9579
	EssenceBench	2.7685	1.6671	1.1516	0.8635	0.7181	0.6101	0.5588	0.5037	0.4561	0.3769
ARC	Random	3.1528	2.4463	2.1646	1.7307	1.4499	1.3194	1.1533	1.0976	0.9341	0.9015
	PPL	5.3343	3.1191	2.1166	1.8187	1.5459	1.4156	1.2165	1.0875	1.0053	0.9168
	GraNd	5.3343	2.9467	2.1475	1.8400	1.6018	1.3504	1.1566	1.0438	1.0004	0.9432
	MetaBench	2.7413	2.0771	1.6838	1.4471	1.2477	1.1103	0.9767	0.9493	0.8626	0.7490
	EssenceBench	2.3990	1.4293	1.1653	0.8023	0.7192	0.6045	0.5053	0.4803	0.4326	0.3699
HellaSwag	Random	2.5738	1.8071	1.4658	1.1984	1.1569	1.0196	0.9751	0.8933	0.7973	0.8342
	PPL	2.9751	1.9949	1.6428	1.2980	1.0798	0.9306	0.9162	0.8333	0.7944	0.7466
	GraNd	2.9223	2.0102	1.5892	1.2572	1.0425	0.9603	0.8844	0.8241	0.8423	0.7825
	MetaBench	2.4339	1.6940	1.4675	1.3135	1.1683	1.0668	0.9926	0.9120	0.8707	0.8220
	EssenceBench	2.2639	1.5717	1.2323	1.0638	0.8906	0.7483	0.6679	0.6150	0.5332	0.5111
WinoGrande	Random	3.4995	2.8713	2.1483	1.9490	1.5938	1.5314	1.2768	1.1537	1.1279	0.9854
	PPL	4.2685	2.7479	2.3403	1.9352	1.7909	1.7748	1.5706	1.4714	1.3994	1.3176
	GraNd	4.2685	2.6562	2.3045	2.0138	1.7775	1.7665	1.6155	1.5049	1.4037	1.2726
	MetaBench	2.7834	2.1219	1.7515	1.5297	1.2893	1.2030	1.0722	0.9578	0.8658	0.7850
	EssenceBench	2.5086	1.3994	0.9791	0.7772	0.6307	0.5580	0.5098	0.4521	0.4134	0.3905
MMLU	Random	3.5048	2.2881	2.1036	1.9096	1.5779	1.5901	1.4984	1.3357	1.3357	1.1699
	PPL	8.0290	9.7627	10.4998	8.5047	8.0146	7.8817	7.6781	7.2748	7.2060	6.7814
	GraNd	8.9996	10.0913	10.4750	8.7332	8.1563	7.9034	7.7453	7.2225	7.5776	6.8507
	MetaBench	2.4268	2.0925	1.7382	1.5292	1.3617	1.2872	1.1992	1.1401	1.0626	0.9941
	EssenceBench	2.4117	1.8293	1.3951	1.1126	1.0220	0.8460	0.7667	0.6906	0.6406	0.5966

constraint. Over successive generations, this process converges toward high-quality subsets that minimize reconstruction error. The whole process is illustrated in Algorithm 1 in Appendix B. Detailed descriptions of each step are provided below.

Individual and Population. An *individual* is a mask $\mathbf{m} \in \{0, 1\}^M$ s.t. $\sum_{j=1}^M m_j = k$, which is also called a *subset*. The *population* is denoted as a group of individuals: $\mathcal{P} = \{\mathbf{m}^{(1)}, \dots, \mathbf{m}^{(N_{\mathcal{P}})}\}$, where $N_{\mathcal{P}}$ is the number of the individuals in the population. The final top- $N_{\mathcal{E}}$ best subsets are also denoted as a set: $\mathcal{E} = \{\mathbf{m}^{(1)}, \dots, \mathbf{m}^{(N_{\mathcal{E}})}\}$.

Fitness Evaluation. The process of *fitness evaluation* is the same as minimizing $\mathcal{L}$ in Equation 2. To calculate, a general additive model (GAM) (Hastie, 2017) is trained to act as the aggregate scoring function g in Definition 2. The training data $\mathcal{T}$ and validation data $\mathcal{V}$ are denoted as $\{s_i, y_i\}_{i \in \mathcal{T}}$ and $\{s_i, y_i\}_{i \in \mathcal{V}}$, which constructs a map from subset accuracy to whole dataset accuracy. Let the score matrix and the accuracy of LLMs on $\mathcal{D}_{\text{filtered}}$ is denoted as: $\mathbf{S}_{\text{filtered}} \in \{0, 1\}^{N_{\text{LLM}} \times M}$, $\mathbf{y}_{\text{filtered}} \in \mathbb{R}^{N_{\text{LLM}}}$. Therefore, y_i can be directly obtained from $\mathbf{y}$, s_i can be calculated as: $s_i = \frac{1}{k} \sum_{j=1}^M \mathbf{S}_{\text{filtered}} \mathbf{m}^{(i)}$. Since RMSE is used as the error measure, given an individual $\mathbf{m}$, its fitness is calculated as:

$$\text{fitness}(\mathbf{m}) = -\text{RMSE}(\mathbf{y}, g(\mathbf{S}_{\mathbf{m}})) = \sqrt{\frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} (\hat{y}_j - y_j)^2}. \quad (9)$$

where $\hat{y}_j = g(s_j)$ is the GAM-predicted accuracy for the j -th individual based on its subset score s_j .

Tournament Selection. In this process, we aim to choose an individual from $\mathcal{P}$ as a *parent* according to fitness (Equation 9). Let $\mathbf{m}^{(a)}, \mathbf{m}^{(b)}$ denote the two parents chosen in this process.

Crossover. In this process, the goal is to get a new individual by combining the information of the two parents. Let ξ_j denote a flag which parent the new individual should follow, the *crossover* process generates a new individual $\mathbf{m}^{(c)}$ by:

$$m_j^{(c)} = (m_j^{(a)} \wedge \xi_j) \vee (m_j^{(b)} \wedge \neg \xi_j), \quad (10)$$

where $\xi_j \sim \text{Bernoulli}(0.5)$, $j \in [1, M]$.

Mutation. In this process, the aim is to introduce randomness into the new individual $\mathbf{m}^{(c)}$, let λ_j be a flag which sample of an individual should mutate:

$$m_j^{(c)} \leftarrow m_j^{(c)} \oplus \lambda_j, \quad (11)$$

where $\lambda_j \sim \text{Bernoulli}(\frac{1}{k})$, $j \in [1, M]$.

Adjustment. To guarantee $\mathbf{m}^{(c)}$ has k -ones, the *adjustment* process is implemented by randomly setting superfluous ones to zeros, vice versa.

Step 3: Attribution-based Sample Selection. To maximize reconstruction fidelity while preserving representational diversity during dataset compression, we train an Explainable Boosting Machine (EBM) (Nori et al., 2019) on the elite mask set $\mathcal{E} = \{\mathbf{m}^{(1)}, \dots, \mathbf{m}^{(N_{\mathcal{E}})}\}$ from Step 2, assigning each sample in $\mathcal{D}_{\text{filtered}} = \{x_1, \dots, x_M\}$ a data-specific attribution score; these scores are used to stratify samples into groups, upon which a genetic algorithm (GA) performs optimized selection to balance signal strength and coverage. The result is a compressed dataset $\mathcal{D}_{\text{compressed}}$ of size $P < M$, which retains high-impact instances while preserving underrepresented patterns — achieving efficient, robust, and generalizable compression without compromising reconstruction performance.

To quantify how much each sample contributes to reconstruction accuracy in the predictor model, we first define the *attribution* of each sample within a mask. For a mask $\mathbf{m} \in \mathcal{E}$, define the selected index set $\mathcal{I}(\mathbf{m}) = \{j \mid m_j = 1\}$. An EBM $g_{\mathbf{m}}(\mathcal{D}_{\text{filtered}}) = \sum_{j \in \mathcal{I}(\mathbf{m})} f_j^{\mathbf{m}}(x_j)$ learns the training data $\mathcal{T}'$. Let $\mathbf{S}_{\text{filtered}, i}$ denote the i -th row of $\mathbf{S}_{\text{filtered}}$, then the form of $\mathcal{T}'$ is: $\{\mathbf{S}_{\text{filtered}, i}, y_i\}_{i \in \mathcal{T}'}$, which constructs a map from sample score matrix to whole dataset accuracy. The component norm $\|f_j^{\mathbf{m}}\|_2$ in $g_{\mathbf{m}}$ is defined as the *attribution* of sample j in a mask. Aggregating over all attributions of each mask in $\mathcal{E}$ yields the *global attribution* of sample j , denoted as A_j :

$$A_j = \frac{\sum_{\mathbf{m} \in \mathcal{E}} \mathbf{1}\{j \in \mathcal{I}(\mathbf{m})\} \|f_j^{\mathbf{m}}\|_2}{\sum_{\mathbf{m} \in \mathcal{E}} \mathbf{1}\{j \in \mathcal{I}(\mathbf{m})\}}, \quad (12)$$

where $j \in \mathcal{I}(\mathcal{D}_{\text{filtered}})$. According to the attributions, a tri-partition of the samples can be implemented. With a retention ratio $\alpha \in (0, 1)$, set $q = \lceil \alpha M \rceil = P \ll M$ and create three groups of equal size q : $G_{\text{high}}, G_{\text{low}}, G_{\text{rand}}$, where G_{high} contains the q samples with the largest A_j , G_{low} the smallest ones, and G_{rand} the random ones. The grouping operation deliberately forces subsequent GAs to search in regions of different attributions so that information neglected by the current top- $N_{\mathcal{E}}$ subsets can be rediscovered, while the information that is really significant can be boosted. For every group $G \in \{G_{\text{high}}, G_{\text{low}}, G_{\text{rand}}\}$, GA is used to judge the best group to get $\mathcal{D}_{\text{compressed}}$.

To increase diversity of pruned dataset, we iteratively repeat Step 2 and Step 3, at each round the globally best mask $\mathbf{m}^*$ is updated whenever a lower error is observed. The details of the entire process are provided in Algorithm 1 in Appendix B.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Baseline Methods. We compared our method with other benchmark compression approaches, including MetaBench. In addition, we also conducted comparisons with several classic methods, including Random Selection, GraNd (Paul et al., 2021), and Perplexity (PPL) (Bengio et al., 2003). The total score of all selected subsets from a given dataset is predicted using a GAM trained on the dataset, and RMSE is computed. The subset with the lowest RMSE is selected as the optimal subset. Please refer to Appendix D.1 for more details.

Dataset Construction. We constructed our dataset using data from the Open LLM Leaderboard (Fourrier et al., 2024) and conducted extensive evaluations of our method and the baselines. The datasets include GSM8K (Cobbe et al., 2021) (1K samples), ARC (Clark et al., 2018) (400 samples), HellaSwag (Zellers et al., 2019) (10K samples), WinoGrande (Sakaguchi et al., 2021) (44K samples), and MMLU (Hendrycks et al., 2020) (15K samples). For data preprocessing, the protocols proposed in MetaBench (Kipnis et al., 2025) were adopted, which involved the removal of low-performing models and items with low variance. For both the training and testing sets, the dataset is first ranked by score and then partitioned into ten equipotent strata. Within each stratum, 10% of the instances are randomly sampled and subsequently pooled to constitute the test set; the remaining 90% are retained as the training set. Please refer to the Appendix D.2 for more details.

4.2 MAIN RESULTS

Better Performance with the Same Compression Ratios. For the results on five benchmarks in Table 1, compared with previous methods, EssenceBench achieves state-of-the-art (SOTA) results

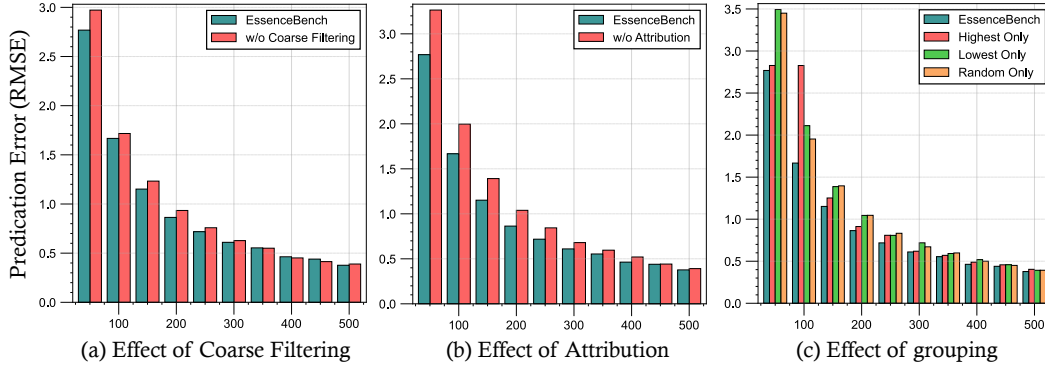


Figure 4: Ablation results on GSM8K, evaluating the effect of (a) coarse filtering, (b) attribution-based selection, and (c) grouping strategies.

across all datasets. Notably, on GSM8K with a subset size of 500, EssenceBench achieves a 60.7% reduction in RMSE compared to MetaBench, demonstrating its consistently superior capability to preserve performance under highly constrained data regimes.

Comparable Performance with Smaller Compression Ratios. EssenceBench achieves comparable or superior performance while using significantly smaller subset sizes, indicating improved data efficiency. For instance, on GSM8K, EssenceBench surpasses the performance of MetaBench using only 200 examples, whereas MetaBench requires 500. A similar trend is observed on the WinoGrande dataset, where EssenceBench outperforms MetaBench with the same reduced subset size of 200, again highlighting its effectiveness under tighter data budgets.

4.3 ABLATION STUDY

Parameter Sensitivity Analysis. To evaluate how hyperparameters influence the effectiveness of EssenceBench, we analyze two factors: (1) the number of generations (*gens*) used in the genetic algorithm to evolve candidate subsets within each iteration, and (2) the number of iterative refinement rounds combining subset search (Step 2) and attribution-based grouping (Step 3), as shown in Figure 3. This analysis aims to reveal the tradeoff between search depth and multi-round refinement in achieving low reconstruction error. Experimental results in Table 2 show that increasing the number of refinement rounds consistently improves performance, regardless of *gens*. For example, with *gens* fixed at 1000, extending from 2 to 5 rounds reduces RMSE from 2.77 to 2.47. When rounds are sufficient (e.g. 5), higher *gens* provides steady improvements, demonstrating that deeper search becomes valuable only when paired with adequate downstream selection. This empirical observation confirms the consistent effectiveness of repeated attribution-guided filtering and recompression in progressively reducing error. On the other hand, increasing *gens* under a small number of rounds (e.g. 2) yields marginal gains or even degrades performance, likely due to over-exploration in a limited refinement context. All results are reported on GSM8K using a fixed 50-sample training set.

Table 2: Performance over Generations and Rounds.

Gen	Round			
	2	3	4	5
1000	2.7685	2.8940	2.7050	2.4731
2000	2.8295	2.7727	2.5338	2.4186
3000	3.0065	2.7494	2.5900	2.4015

The impact of coarse filtering. To isolate the effect of coarse filtering, we compare the full EssenceBench pipeline with a variant that skips redundancy-based filtering and retains only basic outlier removal (Raw EssenceBench). As shown in Figure 4(a), applying coarse filtering significantly reduces RMSE, confirming its effectiveness in removing redundant samples that otherwise degrade reconstruction quality. This performance benefit is most pronounced especially when the subset size is small. As the subset grows, this gain diminishes, likely because the genetic algorithm is more likely to include informative examples regardless of filtering.

The impact of attribution. To investigate the impact of attribution, which is the base of sample selection (Step 3), we compare EssenceBench with merely GA without attribution-based sample selection. As shown in Figure 4(b), when the subset size is below 400, attribution-based sample

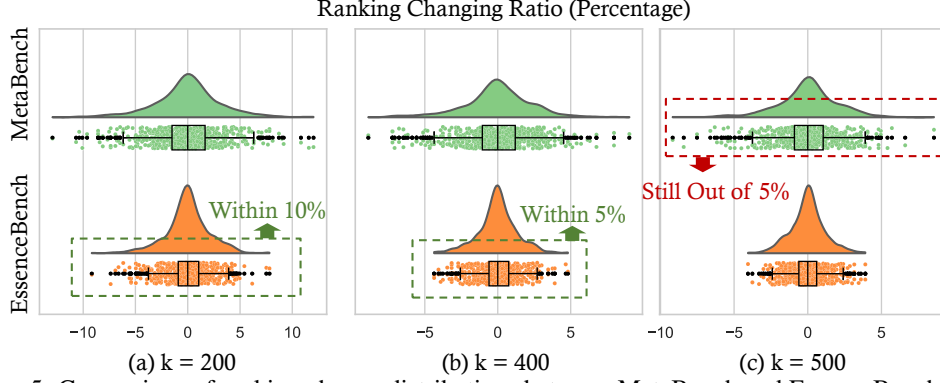


Figure 5: Comparison of ranking change distributions between MetaBench and EssenceBench on the HellaSwag dataset, where k denotes the subset size.

selection outperforms selection without attribution. However, this notable performance advantage largely disappears once the subset size exceeds 400.

The impact of grouping. To assess the role of grouping in Step 3, we compare EssenceBench against three variants: selecting only the top-attribution group (Highest Only), the lowest-attribution group (Lowest Only), or a randomly sampled group (Random Only). All experiments are conducted on GSM8K. As shown in Figure 4(c), EssenceBench consistently outperforms the alternatives when the subset size is small, indicating that attribution-guided diversity effectively enhances compression quality. However, when the subset size exceeds 400, all grouping strategies yield similar RMSE, suggesting diminishing returns from fine-grained selection under larger data budgets.

4.4 CASE STUDY

Text and Ranking Redundancy. To assess the effectiveness of the coarse filtering strategy, we present representative samples with high text and ranking redundancy. As clearly illustrated in Table 3, the proposed filtering mechanism successfully and consistently identifies semantically equivalent items across various cases. In the text redundancy case, two questions differ in phrasing but share the same arithmetic structure. In the ranking redundancy case, two problems with different narratives yield similar model scores because both problems require multi-step numerical reasoning that includes proportional calculations, intermediate variable derivation, and aggregation of weighted quantities. These representative cases provide clear evidence that EssenceBench effectively captures and distinguishes both superficial and deeper structural redundancies.

To assess the effectiveness of the coarse filtering strategy, we present representative samples with high text and ranking redundancy.

Table 3: The most similar text (left) and the highest-ranked similar text (right).

Zack's locker is half as big as Timothy's locker. Peter's locker is 1/4 as big as Zack's locker. If Peter's locker is 5 cubic inches, how big is Timothy's locker in cubic inches?	Axel has 50 silver pesos and 80 gold pesos. He visits her friend Anna who has twice as many silver pesos as he has and 40 more gold pesos. What's the total number of pesos they have together?
Timothy's locker is 24 cubic inches. Zack's locker is half as big as Timothy's locker. Peter's locker is 1/4 as big as Zack's locker. How big is Peter's locker in cubic inches?	Amy is taking a history test. She correctly answers 80% of the multiple-choice questions, 90% of the true/false questions, and 60% of the long-answer questions. The multiple-choice and true/false questions are worth 1 point each, and the long answer questions are worth 5 points each. How many points does Amy score if there are 10 multiple-choice questions, 20 true/false questions, and 5 long answer questions?

Ranking Distribution. To comprehensively evaluate how well the predicted accuracies preserve the original model rankings, we computed several ranking metrics, detailed in the Appendix C and Tables 4, 5, and 6. In particular, we primarily focus on the *ranking error* metric (also referred to as *ranking changing*), which measures the proportion of models whose predicted rank deviates from their true rank by no more than a specified percentage of the total number of models (e.g., within 5% or 10% rank positions). Notably, we found that selecting just $200\times$ less samples preserves 95% of rankings changing within 10%. Furthermore, as shown in Figure 5, EssenceBench consistently outperforms MetaBench in maintaining ranking fidelity. With only 200 samples, all ranking shifts remain within 10%, and with 400 samples, within 5%, offering significantly tighter preservation compared to the deviations observed under MetaBench.

5 CONCLUSION

In this paper, we identified sample redundancy in LLM benchmark evaluation. To address this problem, we introduced EssenceBench, a coarse-to-fine benchmark compression framework that combines redundancy-aware filtering with an iterative genetic algorithm optimized for accurate reconstruction. Extensive experiments on five standard benchmarks show that EssenceBench achieves a $200\times$ reduction in benchmark size while preserving ranking fidelity.

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.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.
- Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024a.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Sijia Chen, Yibo Wang, Yi-Feng Wu, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Lijun Zhang. Advancing tool-augmented large language models: Integrating insights from errors in inference trees. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL <https://openreview.net/forum?id=ZIpdU0cHYu>.
- 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, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. <https://github.com/open-compass/opencompass>, 2023.
- Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.
- Trevor J Hastie. Generalized additive models. *Statistical models in S*, pp. 249–307, 2017.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. 2020.
- Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.

- Alex Kipnis, Konstantinos Voudouris, Luca M. Schulze Buschoff, and Eric Schulz. metabench – a sparse benchmark of reasoning and knowledge in large language models, 2025. URL <https://arxiv.org/abs/2407.12844>.
- Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. *arXiv: Machine Learning, arXiv: Machine Learning*, Mar 2021.
- Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.
- Annu Lambora, Kunal Gupta, and Kriti Chopra. Genetic algorithm-a literature review. In *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*, pp. 380–384. IEEE, 2019.
- Chunyi Li, Xiaozhe Li, Zicheng Zhang, Yuan Tian, Ziheng Jia, Xiaohong Liu, Xionguo Min, Jia Wang, Haodong Duan, Kai Chen, et al. Information density principle for mllm benchmarks. *arXiv preprint arXiv:2503.10079*, 2025.
- Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.
- 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. *arXiv preprint arXiv:2312.15685*, 2023.
- Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.
- Tom V Mathew. Genetic algorithm. *Report submitted at IIT Bombay*, 53:18–19, 2012.
- Seyedali Mirjalili and Seyedali Mirjalili. Genetic algorithm. *Evolutionary algorithms and neural networks: Theory and applications*, pp. 43–55, 2019.
- Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.
- Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13, 2024.
- Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.
- Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34: 20596–20607, 2021.
- Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024.
- 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. *Journal of machine learning research*, 21(140):1–67, 2020.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.
- Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786, 2023.
- 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.
- Wenpeng Yin, Qinyuan Ye, Pengfei Liu, Xiang Ren, and Hinrich Schütze. Llm-driven instruction following: Progresses and concerns. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pp. 19–25, 2023.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.
- Zicheng Zhang, Xiangyu Zhao, Xinyu Fang, Chunyi Li, Xiaohong Liu, Xiongkuo Min, Haodong Duan, Kai Chen, and Guangtao Zhai. Redundancy principles for mllms benchmarks. *arXiv preprint arXiv:2501.13953*, 2025.
- Yiran Zhao, Wenxuan Zhang, Guizhen Chen, Kenji Kawaguchi, and Lidong Bing. How do large language models handle multilingualism? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 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 ADDITIONAL RELATED WORKS

To contextualize our benchmark compression setting within the broader literature, we briefly supplement here the main lines of work on data selection for large language models (LLMs).

Existing methods can be roughly divided into two categories. The first focuses on large-scale data filtering, aiming to eliminate low-quality, toxic, or duplicated samples from pretraining corpora (Raffel et al., 2020; Marion et al., 2023; Zhang et al., 2022; Abbas et al., 2023). These approaches typically prioritize data cleanliness and safety rather than sample representativeness.

The second category explores subset selection strategies that retain model performance with fewer examples. Notably, several works leverage heuristics or Item Response Theory (IRT)-based scoring to select informative subsets (Zhou et al., 2023a; Wang et al., 2023; Ivison et al., 2023). More recent studies also explore the use of LLMs themselves to score or rank data (Xia et al., 2024; Liu et al., 2023), though most of these focus on task-specific learning efficiency rather than benchmark-level score reconstruction or ranking consistency.

Our setting differs in that we explicitly aim to preserve evaluation integrity across LLMs under compression, a goal not typically prioritized in prior work. This highlights the unique focus of our proposed method in balancing compression with benchmarking fidelity.

B PSEUDO CODE OF ESSENCEBENCH

To enhance the reproducibility and clarity of our method, we provide detailed pseudo-code for the core components of EssenceBench. The algorithm consists of two main stages: a single-round Genetic Algorithm for subset selection (Algorithm 1), and a multi-round coarse-to-fine compression framework that iteratively refines sample selection through attribution-based grouping (Algorithm 2).

Algorithm 1: Genetic Algorithm for Subset Selection. This module performs evolutionary search over binary masks that represent subsets of size k from the filtered benchmark. In each generation, the population undergoes fitness evaluation based on RMSE error, followed by tournament selection, crossover, mutation, and adjustment. The top-performing subsets (elites) are retained for both optimization and attribution calculation.

Algorithm 2: Iterative GA with Attribution-Guided Refinement. Building on Algorithm 1, this procedure incorporates sample-level attributions to enhance selection diversity and convergence. In each round, attribution scores are computed from the top- $N_{\mathcal{E}}$ elites using an Explainable Boosting Machine (EBM). Samples are then partitioned into three equally sized groups—High, Low, and Random—and GA is applied to each. The group achieving the lowest error becomes the new candidate pool for the next iteration. The process repeats until convergence or until the size of the pool falls below the desired subset size k .

Key Notations. We briefly summarize the main symbols used throughout our method. The score matrix after coarse filtering is denoted by $\mathbf{S}_{\text{filtered}}$, while $\mathbf{y}$ represents the ground-truth accuracies of all LLMs on the full benchmark. The target coreset size is given by k . In the genetic algorithm (GA) procedure, $N_{\mathcal{P}}$ refers to the population size, and $N_{\mathcal{E}}$ denotes the number of top-performing (elite) candidates retained in each generation. Each GA round runs for N_G generations. The outer coarse-to-fine loop terminates after at most $R_{\max}$ iterations. During attribution-based grouping, α controls the proportion of samples retained in each candidate group, and β denotes the sampling temperature that introduces stochasticity in Step 3.

Algorithm 1: Genetic Algorithm (Subset Selection)

Require: $\mathbf{S}_{\text{filtered}} \in \{0, 1\}^{N_{\text{LLM}} \times M}$, $\mathbf{y} \in \mathbb{R}^{n_{\text{LLM}}}$, k , $N_{\mathcal{P}}$, $N_{\mathcal{E}}$, N_G , $g(\cdot)$
Ensure: Best mask $\mathbf{m}^*$, error ε^* , and final top E subsets $\mathcal{E}$

- 1: Initialize population $\mathcal{P} = \{\mathbf{m}^{(i)}\}_{i=1}^{N_{\mathcal{P}}}$ with random k -masks
- 2: $\mathbf{m}^* \leftarrow \mathbf{0}$, $\varepsilon^* \leftarrow +\infty$
- 3: **for** $t = 1$ to N_G **do**
- 4: **for** each $\mathbf{m} \in \mathcal{P}$ **do**
- 5: Compute accuracy $\hat{s}_i = \frac{1}{k} \mathbf{S}_{\text{filtered}} \mathbf{m}$
- 6: Predict $\hat{y} = g(\hat{s}_i)$
- 7: Compute error $\varepsilon(\mathbf{m}) = \sqrt{\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} (\hat{y}_i - y_i)^2}$
- 8: Set fitness $F(\mathbf{m}) = -\varepsilon(\mathbf{m})$
- 9: **end for**
- 10: Select the top $N_{\mathcal{E}}$ masks by fitness into $\mathcal{E}$
- 11: $\mathcal{P}' \leftarrow \mathcal{E}$
- 12: **while** $|\mathcal{P}'| < N_{\mathcal{P}}$ **do**
- 13: $\mathbf{m}^{(a)}, \mathbf{p}^{(b)} \leftarrow \text{tournament}(\mathcal{P})$
- 14: $\mathbf{m}^{(c)} \leftarrow \text{crossover}(\mathbf{m}^{(a)}, \mathbf{m}^{(b)})$
- 15: $\mathbf{m}^{(c)} \leftarrow \text{mutate}(\mathbf{m}^{(c)})$
- 16: Adjust $\mathbf{m}^{(c)}$ to have exactly k ones
- 17: $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\mathbf{m}^{(c)}\}$
- 18: **end while**
- 19: $\mathcal{P} \leftarrow \mathcal{P}'$
- 20: **if** $\min_{\mathbf{m} \in \mathcal{P}} \varepsilon(\mathbf{m}) < \varepsilon^*$ **then**
- 21: $\varepsilon^* \leftarrow \min_{\mathbf{m}} \varepsilon(\mathbf{m})$
- 22: $\mathbf{m}^* \leftarrow \arg \min_{\mathbf{m}} \varepsilon(\mathbf{m})$
- 23: **end if**
- 24: **end for**
- 25:
- 26: **return** $\mathbf{m}^*, \varepsilon^*, \mathcal{E}$

Algorithm 2: ITERATIVE GA (SUBSET SELECTION + SAMPLE SELECTION)

Require: $\mathbf{S}_{\text{filtered}} \in \{0, 1\}^{n_{\text{LLM}} \times M}$, $\mathbf{y} \in \mathbb{R}^{n_{\text{LLM}} \times 1}$, k , R_{max} , α , β
Ensure: best mask $\mathbf{m}^*$, error ε^*

- 1: $\mathcal{I} \leftarrow \{1, \dots, M\}$, $\mathbf{m}^* \leftarrow \mathbf{0}$, $\varepsilon^* \leftarrow \infty$
- 2: **for** $r = 0$ to $R_{\text{max}} - 1$ **do**
- 3: $(\mathbf{m}, \varepsilon, \mathcal{E}) \leftarrow \text{GA_SEARCH}(\mathbf{S}(:, \mathcal{I}), \mathbf{y}, k)$
- 4: **if** $\varepsilon < \varepsilon^*$ **then**
- 5: update $(\mathbf{m}^*, \varepsilon^*)$
- 6: **end if**
- 7: **if** $|\mathcal{I}| \leq k$ **or** $r = R_{\text{max}} - 1$ **then**
- 8: **break**
- 9: **end if** {attributions aggregation}
- 10: compute attributions $\{A_j\}_{j \in \mathcal{I}}$ from the top- $N_{\mathcal{E}}$ subsets $\mathcal{E}$
- 11: sort A_j ; let $q = \lceil \alpha \cdot |\mathcal{I}| \rceil$
- 12: $G_{\text{high}} \leftarrow$ top- q samples, $G_{\text{low}} \leftarrow$ bottom- q samples, $G_{\text{rand}} \leftarrow$ random- q samples
- 13: **for** $G \in \{G_{\text{high}}, G_{\text{low}}, G_{\text{rand}}\}$ **do**
- 14: $(-, \varepsilon_G, -) \leftarrow \text{GA_SEARCH}(\mathbf{S}(:, G), \mathbf{y}, k; \beta)$
- 15: **end for**
- 16: $\mathcal{I} \leftarrow \arg \min_G \varepsilon_G$
- 17: **end for**
- 18:
- 19: **return** $\mathbf{m}^*, \varepsilon^*$

C DETAILS OF RANKING DISTRIBUTION

Metrics. Each table compares MetaBench and EssenceBench on eight well-defined statistics that together quantify how faithfully a small subset reproduces the full leaderboard.

RMSE ($\downarrow$). Let y_i be the true accuracy of model i on the full benchmark, and $\hat{y}_i$ the accuracy estimated from the compressed subset. With n models,

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

so a lower value indicates more accurate prediction of overall scores.

Rank correlations ($\uparrow$). We report three correlation metrics: Pearson’s r , Spearman’s ρ , and Kendall’s τ . Pearson correlation is computed on raw scores (y_i) and ($\hat{y}_i$), reflecting linear agreement. Spearman and Kendall are based on rankings $\text{rank}(y_i)$ and $\text{rank}(\hat{y}_i)$, capturing monotonic consistency even when score magnitudes differ.

Rank Stability ($\uparrow$). We define average positional deviation $\delta_i = |\text{rank}(y_i) - \text{rank}(\hat{y}_i)|$, normalised over all models:

$$\text{Stability} = 1 - \frac{1}{n} \sum_{i=1}^n \frac{\delta_i}{n}.$$

This ranges from 1 (perfect ranking match) to 0 (completely disordered).

Pair Accuracy ($\uparrow$). This measures the fraction of model pairs that preserve their relative ranking:

$$\text{PairAcc} = \frac{1}{\binom{n}{2}} \sum_{i < j} \mathbf{1}[(y_i > y_j) \Leftrightarrow (\hat{y}_i > \hat{y}_j)].$$

Top-tier retrieval ($\uparrow$). NDCG@50 is the Normalized Discounted Cumulative Gain over the top-50 predicted models, computed using the ground-truth order as ideal ranking. Top-50 Accuracy measures the intersection-over-union between true and predicted top-50 sets.

Ranking Error within $\{1, 2, 5, 10\}\%$ ($\uparrow$). For a given tolerance $p \in \{1, 2, 5, 10\}$, this reports the proportion of models whose predicted rank deviates from the ground-truth rank by at most $\lceil pn/100 \rceil$ positions. Higher values mean better rank preservation under tighter constraints.

Together, these metrics assess both absolute score accuracy and ranking quality, from overall correlation down to local ordering and top-model retrieval fidelity.

Table organisation. Each block of rows corresponds to the metrics just defined, while the columns enumerate subset sizes from 50 to 500 test items. Within every cell the lower (for RMSE) or higher (for the seven ranking metrics) value is emboldened so that the superior method-MetaBench or EssenceBench-is visible at a glance.

Key findings. On all three representative benchmarks EssenceBench achieves the lowest RMSE, with the margin especially large when only 50–250 examples are retained. Correlation metrics (Pearson, Spearman, Kendall) and top-50 retrieval scores are likewise higher for EssenceBench, indicating much closer alignment to the full-set leaderboard. In GSM8K and ARC, for example, EssenceBench attains the same ranking stability with roughly 150–200 samples that MetaBench needs 400–500 samples to reach, underscoring the efficiency of our coarse-to-fine search.

Comprehensive numbers are reported in Tables 4, 5 and 6.

D DETAILS OF EXPERIMENTS

D.1 BASELINE METHODS FOR DATA SELECTION

For baseline comparisons, we include the following widely used methods. Random Selection selects subsets by randomly sampling from the original data. GraNd selects the top-k data points based on

Table 4: MetaBench vs. EssenceBench on GSM8K dataset (↑ larger is better, ↓ smaller is better)

Method & Metric	GSM8K Coreset Size									
	50	100	150	200	250	300	350	400	450	500
MetaBench										
RMSE↓	3.508	2.563	2.053	1.765	1.535	1.388	1.209	1.146	1.047	0.960
Pearson↑	0.991	0.995	0.997	0.998	0.998	0.999	0.999	0.999	0.999	0.999
Spearman↑	0.984	0.989	0.994	0.994	0.996	0.996	0.997	0.997	0.998	0.998
Kendall↑	0.888	0.912	0.934	0.937	0.949	0.950	0.956	0.960	0.964	0.967
Rank Stability↑	0.018	0.028	0.033	0.033	0.041	0.039	0.043	0.049	0.056	0.059
Pair Accuracy↑	0.930	0.948	0.960	0.963	0.970	0.971	0.975	0.977	0.979	0.981
NDCG@50↑	0.976	0.990	0.995	0.991	0.994	0.995	0.996	0.995	0.998	0.998
Top50 Accuracy↑	0.720	0.820	0.880	0.840	0.880	0.900	0.860	0.840	0.920	0.940
Ranking Error within 1%↑	0.185	0.245	0.298	0.358	0.386	0.406	0.488	0.517	0.504	0.597
Ranking Error within 2%↑	0.357	0.457	0.556	0.596	0.660	0.681	0.741	0.755	0.786	0.841
Ranking Error within 5%↑	0.684	0.794	0.887	0.897	0.941	0.943	0.953	0.966	0.984	0.984
Ranking Error within 10%↑	0.953	0.959	0.987	0.984	0.997	0.993	0.998	0.998	1.000	1.000
EssenceBench										
RMSE↓	2.900	1.525	1.131	0.857	0.682	0.597	0.543	0.454	0.426	0.375
Pearson↑	0.994	0.998	0.999	0.999	1.000	1.000	1.000	1.000	1.000	1.000
Spearman↑	0.987	0.995	0.997	0.998	0.999	0.999	0.999	0.999	0.999	1.000
Kendall↑	0.905	0.943	0.955	0.968	0.973	0.976	0.978	0.980	0.982	0.985
Rank Stability↑	0.023	0.033	0.046	0.074	0.097	0.093	0.100	0.116	0.128	0.165
Pair Accuracy↑	0.939	0.963	0.972	0.979	0.983	0.984	0.986	0.987	0.988	0.990
NDCG@50↑	0.983	0.993	0.996	0.998	0.998	0.999	0.999	1.000	0.999	1.000
Top50 Accuracy↑	0.780	0.860	0.860	0.940	0.880	0.940	0.900	0.920	0.940	0.920
Ranking Error within 1%↑	0.237	0.383	0.457	0.609	0.661	0.727	0.730	0.792	0.813	0.864
Ranking Error within 2%↑	0.416	0.637	0.727	0.822	0.879	0.902	0.912	0.923	0.944	0.964
Ranking Error within 5%↑	0.774	0.917	0.946	0.982	0.990	0.990	0.995	0.997	0.995	1.000
Ranking Error within 10%↑	0.956	0.989	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 5: MetaBench vs. EssenceBench on HellaSwag dataset (↑ larger is better, ↓ smaller is better)

Method & Metric	HellaSwag Coreset Size									
	50	100	150	200	250	300	350	400	450	500
MetaBench										
RMSE↓	2.397	1.734	1.495	1.318	1.166	1.076	0.961	0.929	0.880	0.836
Pearson↑	0.992	0.996	0.997	0.998	0.998	0.998	0.999	0.999	0.999	0.999
Spearman↑	0.979	0.989	0.992	0.994	0.995	0.996	0.997	0.997	0.997	0.997
Kendall↑	0.879	0.915	0.928	0.937	0.942	0.946	0.955	0.954	0.957	0.959
Rank_Stability↑	0.024	0.024	0.021	0.030	0.045	0.032	0.054	0.044	0.062	0.042
Pair_Accuracy↑	0.921	0.949	0.957	0.963	0.967	0.970	0.974	0.974	0.976	0.977
NDCG@50↑	0.990	0.996	0.997	0.997	0.997	0.997	0.998	0.999	0.999	0.999
Top50_Accuracy↑	0.920	0.940	0.960	0.940	0.900	0.960	0.900	0.960	0.940	0.980
Ranking_Error_within 1%↑	0.200	0.265	0.278	0.355	0.377	0.395	0.418	0.435	0.492	0.466
Ranking_Error_within 2%↑	0.370	0.487	0.537	0.586	0.621	0.629	0.705	0.683	0.717	0.725
Ranking_Error_within 5%↑	0.660	0.812	0.854	0.889	0.905	0.928	0.959	0.949	0.961	0.974
Ranking_Error_within 10%↑	0.902	0.961	0.983	0.989	0.991	1.000	0.998	1.000	0.998	1.000
EssenceBench										
RMSE↓	2.153	1.453	1.038	0.863	0.706	0.635	0.504	0.461	0.409	0.419
Pearson↑	0.994	0.997	0.999	0.999	0.999	0.999	1.000	1.000	1.000	1.000
Spearman↑	0.985	0.993	0.997	0.997	0.998	0.998	0.999	0.999	0.999	0.999
Kendall↑	0.899	0.931	0.951	0.959	0.966	0.967	0.973	0.974	0.978	0.978
Rank_Stability↑	0.032	0.050	0.062	0.062	0.068	0.069	0.107	0.095	0.129	0.102
Pair_Accuracy↑	0.932	0.958	0.970	0.975	0.980	0.981	0.984	0.985	0.987	0.987
NDCG@50↑	0.992	0.996	0.996	0.998	0.999	0.999	0.999	0.999	0.999	0.999
Top50_Accuracy↑	0.960	0.920	0.940	0.940	0.940	0.980	0.960	0.960	0.960	0.980
Ranking_Error_within 1%↑	0.257	0.337	0.392	0.492	0.520	0.552	0.620	0.645	0.666	0.677
Ranking_Error_within 2%↑	0.432	0.535	0.645	0.725	0.788	0.818	0.865	0.871	0.910	0.913
Ranking_Error_within 5%↑	0.738	0.869	0.964	0.974	0.989	0.988	0.997	1.000	1.000	1.000
Ranking_Error_within 10%↑	0.946	0.988	0.998	1.000	1.000	1.000	1.000	1.000	1.000	1.000

the gradient norms of the final token in the prediction task, ranking samples in descending order of gradient magnitude. PPL (Perplexity) is a standard metric for evaluating language models, defined as the exponential of the average negative log-likelihood over the predicted tokens, and is computed over the full predicted sequence for each question in the dataset. To compute the gradients for GraNd-based selection, we use the Llama-3.1-8B-Instruct³ model as the scoring backbone.

³<https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>

Table 6: MetaBench vs. EssenceBench on ARC dataset ($\uparrow$ larger is better, $\downarrow$ smaller is better)

Method & Metric	ARC Coreset Size									
	50	100	150	200	250	300	350	400	450	500
MetaBench										
RMSE $\downarrow$	2.968	2.082	1.690	1.511	1.332	1.186	1.077	0.961	0.890	0.836
Pearson $\uparrow$	0.986	0.993	0.995	0.996	0.997	0.998	0.998	0.998	0.999	0.999
Spearman $\uparrow$	0.976	0.984	0.991	0.992	0.994	0.995	0.996	0.997	0.997	0.998
Kendall $\uparrow$	0.870	0.898	0.922	0.928	0.936	0.942	0.949	0.955	0.959	0.961
Rank_Stability $\uparrow$	0.014	0.021	0.026	0.027	0.041	0.035	0.038	0.050	0.044	0.044
Pair_Accuracy $\uparrow$	0.918	0.940	0.954	0.959	0.963	0.967	0.972	0.974	0.977	0.978
NDCG@50 $\uparrow$	0.987	0.992	0.995	0.994	0.997	0.998	0.998	0.999	0.999	0.998
Top50_Accuracy $\uparrow$	0.900	0.880	0.900	0.900	0.880	0.880	0.920	0.920	0.900	0.960
Ranking_Error_within 1% $\uparrow$	0.150	0.236	0.274	0.317	0.368	0.361	0.412	0.459	0.456	0.492
Ranking_Error_within 2% $\uparrow$	0.328	0.442	0.492	0.552	0.597	0.615	0.669	0.704	0.753	0.767
Ranking_Error_within 5% $\uparrow$	0.653	0.756	0.844	0.853	0.887	0.902	0.952	0.956	0.974	0.970
Ranking_Error_within 10% $\uparrow$	0.881	0.929	0.973	0.985	0.985	0.992	0.998	1.000	1.000	0.998
EssenceBench										
RMSE $\downarrow$	2.045	1.104	0.841	0.703	0.612	0.529	0.501	0.422	0.368	0.347
Pearson $\uparrow$	0.993	0.998	0.999	0.999	0.999	1.000	1.000	1.000	1.000	1.000
Spearman $\uparrow$	0.988	0.996	0.997	0.998	0.998	0.999	0.999	0.999	0.999	0.999
Kendall $\uparrow$	0.907	0.947	0.958	0.964	0.969	0.973	0.974	0.978	0.981	0.983
Rank_Stability $\uparrow$	0.035	0.047	0.063	0.071	0.077	0.096	0.084	0.120	0.147	0.149
Pair_Accuracy $\uparrow$	0.939	0.965	0.973	0.977	0.980	0.983	0.983	0.986	0.988	0.989
NDCG@50 $\uparrow$	0.986	0.995	0.999	0.999	0.999	0.999	1.000	0.999	1.000	0.999
Top50_Accuracy $\uparrow$	0.900	0.940	0.960	0.920	0.960	1.000	0.960	0.980	0.960	0.940
Ranking_Error_within 1% $\uparrow$	0.233	0.408	0.444	0.498	0.550	0.630	0.639	0.696	0.737	0.771
Ranking_Error_within 2% $\uparrow$	0.433	0.641	0.735	0.768	0.823	0.869	0.871	0.901	0.937	0.955
Ranking_Error_within 5% $\uparrow$	0.788	0.928	0.965	0.986	0.991	0.997	0.997	0.998	1.000	1.000
Ranking_Error_within 10% $\uparrow$	0.949	0.997	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

D.2 DETAILS OF DATASET CONSTRUCTION

Following the initial collection of benchmark data, we performed a coarse filtering step to remove redundant or uninformative items before subset selection. This stage was guided by two primary criteria: textual similarity and ranking consistency.

To mitigate redundancy in the benchmark, we applied coarse filtering using both semantic and behavioral criteria. For semantic overlap, we used the `bge-m3` model (Chen et al., 2024a) to embed each item and computed pairwise similarities across all examples. Items with high embedding similarity were removed to ensure lexical and conceptual diversity. In parallel, we identified items with redundant behavioral signals by examining their ranking patterns over LLMs. Items that yielded highly correlated rankings were pruned, as they offered limited additional insight into model differences. Together, these two filters reduced both surface-level and functional redundancy, yielding a more informative candidate pool for downstream compression.

In the case of the MMLU dataset, we observed that many items exhibited naturally high textual and behavioral similarity due to the curriculum-style structure of the benchmark. To avoid over-pruning, we adapted the filtering thresholds to be more lenient for MMLU.