From Tags to Trees: Structuring Fine-Grained Knowledge for Controllable Data Selection in Instruction Tuning

Anonymous ACL submission

Abstract

Effective and controllable data selection is critical for instruction tuning, especially with massive open-source datasets. Existing approaches primarily rely on instance-level quality scores, or diversity metrics based on embedding clusters or semantic tags. However, constrained by the flatness of embedding spaces or the coarseness of tags, these approaches overlook fine-grained knowledge and its intrinsic hierarchical dependencies, consequently hindering precise data valuation and knowledge-aligned sampling. To address this challenge, we propose Tree-aware Aligned Global Sampling (TAGS), a unified framework that leverages a knowledge tree built from fine-grained tags, thereby enabling joint control of global quality, diversity, and target alignment. Using an LLM-based tagger, we extract atomic knowledge concepts, which are organized into a global tree through bottom-up hierarchical clustering. By grounding data instances onto this tree, a tree-aware metric then quantifies data quality and diversity, facilitating effective sampling. Our controllable sampling strategy maximizes tree-level information gain and enforces leaf-level alignment via KL-divergence for specific domains. Extensive experiments demonstrate that TAGS significantly outperforms state-of-the-art baselines. Notably, it surpasses the full-dataset model by **+5.84%** using only **5%** of the data, while our aligned sampling strategy further boosts average performance by **+4.24%**.

1 Introduction

Large Language Models (LLMs) have become a central paradigm in natural language processing, achieving strong performance across a wide range of tasks (Yang et al., 2025; Liu et al., 2024a; Achiam et al., 2023). While pre-training provides broad world knowledge, instruction tuning plays a crucial role in aligning their behaviors with human intent (Ouyang et al., 2022; Wei et al., 2021). Consequently, selecting high-quality, diverse, and task-relevant instruction data has become essential for effective instruction tuning.

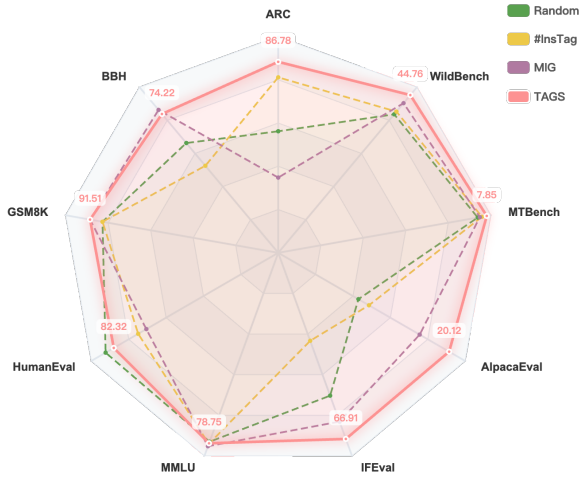

Figure 1: Performance comparison of Qwen3-8B fine-tuned on Tulu3. TAGS consistently outperforms other data selection strategies.

The rapid growth of open-source instruction datasets (Lambert et al., 2024; Teknium, 2023; Xu et al., 2024) has provided abundant resources for instruction tuning, making it increasingly important to identify high-value data from large candidate pools. However, more data does not necessarily lead to better models. Recent studies (Zhou et al., 2023a; Ge et al., 2024; Chen et al., 2025) reveal a *"less is more"* phenomenon: models trained on carefully selected subsets can outperform those trained on much larger corpora. This has shifted attention from indiscriminate data scaling to automated data selection, which aims to reduce training cost while avoiding redundant, biased, or noisy instructions that may cause an *alignment tax* (Ouyang et al., 2022; Zhou et al., 2023a).

Existing automated selection methods mainly optimize data quality, semantic diversity, or their combination. Metric-based filtering methods (Chen et al., 2023; Li et al., 2024a; Liu et al., 2024b) score

each instance independently, but overlook subset-level diversity. Diversity-driven methods (Yin et al., 2024; Bukharin et al., 2024) improve semantic coverage, yet typically rely on flat embedding spaces that fail to preserve fine-grained conceptual relations. More comprehensive methods (Lu et al., 2023; Ge et al., 2024; Chen et al., 2025) combine quality and diversity through tagging or clustering, but often depend on coarse or weakly structured labels, making it difficult to distinguish atomic concepts such as *"matrix decomposition"* and *"linear equations"* under broad categories like *"Math"*. Overall, existing methods lack an explicit fine-grained knowledge topology, making it difficult to estimate the marginal value of candidate samples and to control the selected distribution toward target-domain requirements.

To address these limitations, we propose TAGS (**T**ree-aware **A**ligned **G**lobal **S**ampling), a unified framework that introduces a fine-grained knowledge topology for controllable instruction data selection. TAGS first employs a lightweight tagger to extract atomic knowledge concepts from large-scale instruction data (Teknium, 2023; Xu et al., 2024, 2025; LI et al., 2024; Zheng et al., 2023a). These concepts are then organized into a global Tag Tree through bottom-up hierarchical clustering, where leaf nodes correspond to specific knowledge units and higher-level nodes capture broader semantic categories. Based on this tree, TAGS anchors each instruction instance to relevant knowledge nodes and estimates its contribution through tree-aware information propagation. A concave aggregation function further imposes diminishing returns on repeated concepts, thereby reducing redundancy while preserving global knowledge coverage. Finally, TAGS supports two sampling strategies: **General Sampling**, which maximizes global information gain, and **Aligned Sampling**, which aligns the selected data distribution with target-domain requirements via KL-divergence.

We conduct main experiments on the Qwen3 model family (Yang et al., 2025) across multiple scales, using a broad set of knowledge-intensive (Clark et al., 2018; Suzgun et al., 2023; Hendrycks et al., 2020; Chen, 2021; Cobbe et al., 2021; Zhou et al., 2023b) and human-preference benchmarks (Zheng et al., 2023b; Dong et al., 2024; Lin et al., 2024). As shown in Figure 1, TAGS consistently outperforms strong data selection baselines. Notably, with only 5% of the original data, TAGS surpasses full-dataset fine-tuning by 2.23% on knowledge-intensive benchmarks and 12.34% on human-preference evaluations. In addition, Knowledge-Aligned Sampling improves target-domain performance by 4.24% over the general strategy, demonstrating the effectiveness of tree-aware controllable selection.

In summary, our contributions are as follows:

- We propose TAGS, a tree-aware instruction data selection framework that introduces fine-grained knowledge structures to jointly capture data quality, diversity, and target alignment.
- We design two sampling strategies, General Sampling and Aligned Sampling, enabling both broad-coverage selection and controllable adaptation to target-domain requirements.
- Extensive experiments on Qwen3 models demonstrate that TAGS achieves state-of-the-art data efficiency, with only 5% selected data outperforming full-dataset fine-tuning. Additional results on Mistral and LLaMA backbones further demonstrate the generality of TAGS.

2 Related Work

Instruction Tuning and Data Efficiency. Instruction tuning is widely recognized as an important approach for aligning LLMs with human intent (Ouyang et al., 2022; Wei et al., 2021; Sanh et al., 2021). Recent studies (Zhou et al., 2023a; Ge et al., 2024; Chen et al., 2025) show that carefully curated instruction subsets can outperform much larger training corpora during Supervised Fine-Tuning (SFT). These findings highlight the importance of strategic data selection, where automated methods are required to identify high-value samples while balancing data quality and diversity.

Automated Data Selection. Existing automated data selection methods (Chen et al., 2023; Lu et al., 2023; Ge et al., 2024; Liu et al., 2024b; Chen et al., 2025) aim to identify compact training subsets that improve both efficiency and performance. Broadly, they can be categorized into metric-based filtering, diversity-driven selection, and comprehensive quality-diversity strategies. (1) ***Metric-based filtering*** relies on scalar criteria to assess data quality, difficulty, or uncertainty. IFD (Li et al., 2024a) estimates instruction difficulty through the discrepancy between predicted and self-generated responses, while ALPAGASUS (Chen et al., 2023) employs LLM-based judges and SelectIT (Liu et al., 2024b) filters samples based on intrinsic uncertainty. Although effective for denoising, these

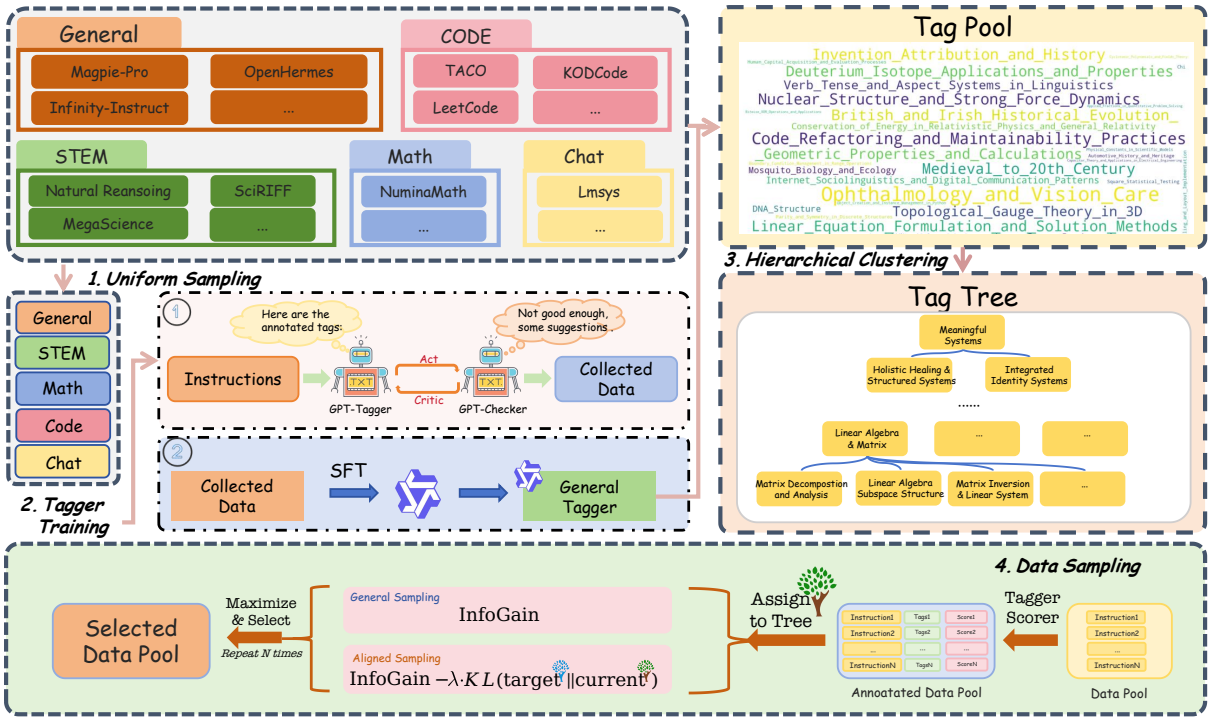

Figure 2: Overview of the TAGS framework: (1) **Uniform Sampling** from diverse data sources; (2) **Tagger Training**, which employs an LLM-based Act-Critic loop to synthesize fine-grained tags and fine-tune a General Tagger; (3) **Hierarchical Clustering** structures the generated tag pool into a global Tag Tree; and (4) **Data Sampling** maps instances to tree nodes and executes controllable selection based on Information Gain and KL-divergence.

methods evaluate instances independently and often ignore subset-level diversity. (2) ***Diversity-driven selection*** emphasizes coverage of the semantic space. ZIP (Yin et al., 2024) promotes diversity through a compression-based metric, whereas QDIT (Bukharin et al., 2024) selects representative samples that maximize current subset diversity. However, these methods typically measure diversity in flat representation spaces, making it difficult to preserve fine-grained conceptual structures. (3) ***Comprehensive strategies*** seek to jointly balance quality and diversity during selection. CaR (Ge et al., 2024) performs clustering followed by intra-cluster quality selection, while DEITA (Liu et al., 2024c) greedily selects high-quality samples with semantic redundancy penalties. *#InsTag* (Lu et al., 2023) introduces LLM-generated coarse-grained tags and selects data based on tag coverage and complexity, while MIG (Chen et al., 2025) further models semantic relations as a label graph and maximizes information gain. Despite these advances, existing methods largely overlook fine-grained knowledge concepts and their hierarchical dependencies, limiting accurate sample utility estimation and target-domain alignment. To address this issue, we introduce TAGS, which leverages fine-grained tags and tree-structured knowledge topology for quality-aware, diversity-preserving, and knowledge-aligned instruction data selection.

3 Method

As shown in Figure 2, we propose TAGS, a tree-aware data selection framework that enables both general and aligned sampling. TAGS extracts fine-grained knowledge tags with a trained LLM tagger (Sec. 3.2), organizes them into a hierarchical Tag Tree (Sec. 3.3), and selects data through tree-aware information gain and KL-based distribution alignment (Sec. 3.4).

3.1 Problem Formulation & Notation

Task. Given a data budget N, a data pool D_P, and a large language model M, our goal is to select a subset $D_S \subset D_P$ with $|D_S| = N$ such that the performance of M fine-tuned on D_S, denoted by $P(M, D_S)$, is maximized:

$$D_S = \underset{D \subset D_P, |D|=N}{\operatorname{argmax}} \; P(M, D). \qquad (1)$$

Since directly optimizing this objective requires repeatedly fine-tuning and evaluating M, TAGS

approximates it from a data-centric perspective by estimating the utility and coverage of candidate samples without querying the target model during selection.

Data. Each data instance $d_i \in D_P$ consists of a query-response pair (q_i, r_i) and associated attributes (T_i, Q_i, C_i), where q_i and r_i denote the query and response, respectively. T_i denotes the extracted fine-grained knowledge tags, while Q_i and C_i denote the quality and complexity scores of d_i, respectively. We obtain Q_i and C_i using the quality and complexity scorers from DEITA (Liu et al., 2024c), which are later incorporated into the sampling objective.

3.2 Fine-grained Tagger Training

Existing approaches (Ge et al., 2024; Bukharin et al., 2024) often use embeddings and clustering to estimate data distributions and diversity. However, such latent representations lack interpretability, making it difficult to determine whether clusters arise from shared linguistic patterns, core knowledge, or other latent factors. In contrast, explicit semantic tags map unstructured instructions into a lower-noise, more controllable semantic space. Nevertheless, as illustrated in Figure 3, existing taggers such as *#InsTag* mainly capture coarse-grained intents and struggle to identify specific entities, contexts, constraints, and knowledge concepts embedded in complex instructions. This motivates training a fine-grained tagger for reliable knowledge concept extraction.

In TAGS, each instruction is assigned multiple fine-grained tags, each denoting an atomic knowledge concept such as a mathematical operation, programming concept, or domain-specific entity. We construct a high-quality training corpus for fine-grained tagging, with 50k samples spanning core domains and an additional 3k manually curated complex instructions to improve the tagger's ability to handle challenging queries (see Table 6 for details). Inspired by the Self-Refine mechanism (Madaan et al., 2023), we introduce an Act-Critic annotation pipeline powered by `GPT-o4-mini`. In this pipeline, a *GPT-Tagger* first generates candidate tags, while a *GPT-Checker* reviews them and provides refinement suggestions. To balance annotation quality and efficiency, we limit refinement to at most three iterations. Finally, we perform SFT on `Qwen3-8B-Instruct` (Yang et al., 2025) using the annotated corpus, yielding an efficient tagger for large-scale data.

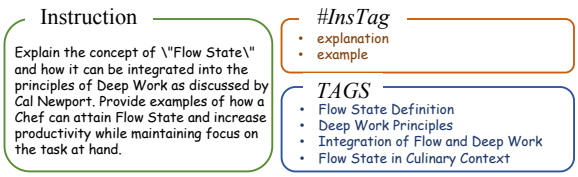

Figure 3: Comparison of *#InsTag* and TAGS tagger.

3.3 Hierarchical Tag Tree Construction

Although the TAGS tagger extracts fine-grained knowledge concepts, the resulting tag pool remains fragmented and structurally flat. Similar concepts may be assigned different surface forms, such as *"Value-to-Index Mapping"* and *"Index-based count computation"*, while hierarchical relations among concepts, such as *"Matrix Decomposition"* under *"Linear Algebra"*, are not explicitly represented. This lack of structural organization makes it difficult to assess global data coverage and redundancy.

To address this issue, we construct a Hierarchical Tag Tree that organizes scattered fine-grained tags into an interpretable knowledge topology for downstream coverage estimation and sampling. The tree groups semantically related tags into higher-level concepts, reducing surface-level fragmentation while preserving fine-grained distinctions. Inspired by Clio (Tamkin et al., 2024), we adopt an iterative bottom-up construction framework. Starting from fine-grained tags as leaf nodes, the framework progressively abstracts them into broader semantic categories, which naturally reflects the hierarchical organization of knowledge. At each iteration, we first cluster the current nodes using K-Means over semantic embeddings[1]. Since embedding-based clustering is efficient but may introduce noisy or ambiguous groupings, we further employ an LLM[2] to refine the clusters through a structured four-step pipeline:

- **Summarization:** generating a representative topic for each cluster;
- **Deduplication:** merging synonymous or highly overlapping topics;
- **Re-assign:** reviewing member tags and moving misclassified items to more appropriate topics;
- **Renaming:** refining each topic name to better reflect its corrected cluster composition.

The refined topics are treated as higher-level nodes and used as inputs for the next iteration. This process is repeated until a predefined tree depth is reached, resulting in a global Tag Tree where leaf

[1] We utilize `Qwen3-Embedding-0.6B` as the encoder.
[2] `Qwen3-30B-A3B-2507`.

nodes correspond to fine-grained knowledge concepts and internal nodes represent progressively broader semantic categories. We use a 10-layer Tag Tree in our main experiments, with the effect of tree depth analyzed in Appendix B.6.

3.4 Tree-aware Data Sampling

Tree Topology & Data Anchoring. Let $\mathcal{T} = (V, E)$ denote the Tag Tree constructed in Sec. 3.3, where V is the set of nodes and E encodes hierarchical relations. We denote the leaf nodes by $V_{\text{leaf}} \subset V$, where each leaf represents an atomic knowledge concept. To incorporate the tree hierarchy, we define an ancestry matrix $\mathbf{M} \in \{0,1\}^{|V| \times |V_{\text{leaf}}|}$:

$$\mathbf{M}_{ij} = \mathbb{I}\left(i \in \text{Anc}(j) \cup \{j\}\right), \tag{2}$$

where $i \in V$, $j \in V_{\text{leaf}}$, $\text{Anc}(j)$ denotes the ancestors of leaf node j, and $\mathbb{I}(\cdot)$ is the indicator function. For each sample d_k, we map its extracted tags T_k to the Tag Tree by matching each tag to its nearest leaf node in the embedding space according to semantic similarity. The matched leaves form a binary activation vector $\mathbf{h}_k^{(\text{leaf})} \in \{0,1\}^{|V_{\text{leaf}}|}$, where an entry is activated if the corresponding leaf is matched by at least one tag in T_k. We then lift this leaf-level representation to the full tree:

$$\mathbf{h}_k^{(\text{tree})} = \mathbf{M}\mathbf{h}_k^{(\text{leaf})}. \tag{3}$$

The resulting profile captures both atomic concepts and higher-level semantic categories in d_k.

Tree-based Information Modeling. Given the tree-level profile $\mathbf{h}_k^{(\text{tree})}$, we estimate the utility of sample d_k by combining its quality score Q_k and complexity score C_k:

$$s_k = \alpha Q_k + (1-\alpha) C_k, \quad \mathbf{e}_k = s_k \cdot \mathbf{h}_k^{(\text{tree})}, \tag{4}$$

where $\alpha \in [0,1]$ controls the trade-off between the two scores, and $\mathbf{e}_k$ denotes the raw information vector of d_k. To capture semantic dependencies among hierarchical concepts, we propagate information along the tag tree. Following Chen et al. (2025), the propagation coefficient from node p to node q is defined as:

$$\mathbf{A}_{pq} = \frac{w_{pq}}{1 + \sum_{j \neq p} w_{pj}}, \tag{5}$$

where $w_{pq} = \mathbb{I}((p,q) \in E)$ indicates whether nodes p and q are structurally connected. This allows sample coverage to be measured over related conceptual branches instead of isolated nodes.

With the propagation matrix $\mathbf{A}$, the raw information of sample d_k is transformed into $\mathbf{A}\mathbf{e}_k$. For a selected subset D, we first aggregate the propagated information over all selected samples, and then apply a nonlinear activation to each node before summing across the tag tree:

$$\mathcal{I}(D) = \sum_{j \in V} \Phi(\mathbf{A} \sum_{k \in D} \mathbf{e}_k), \tag{6}$$

where $\Phi(x) = x^\gamma$ with $\gamma \in (0,1)$ is strictly increasing and concave, favoring high-utility samples while introducing diminishing returns to reduce redundancy and balance quality with diversity.

Tree-aware Dual-mode Sampling. To curate a high-quality subset, we employ a greedy selection strategy based on marginal information gain. Since exact re-evaluation of the objective for each candidate is computationally costly, we approximate the marginal gain via a first-order Taylor expansion:

$$\begin{aligned} d_k &= \operatorname*{argmax}_{d \in D_P^k} \left(\mathcal{I}(D_S^k \cup \{d\}) - \mathcal{I}(D_S^k) \right) \\ &\approx \operatorname*{argmax}_{d \in D_P^k} \left(\nabla_{\mathbf{v}} \mathcal{I}(\mathbf{v}_k)^\top (\mathbf{A}\mathbf{e}_d) \right) \\ &= \operatorname*{argmax}_{d \in D_P^k} (\Phi'(\mathbf{A} \sum_{i \in D_S^k} \mathbf{e}_i)^\top \mathbf{A}\mathbf{e}_d), \end{aligned} \tag{7}$$

where $\mathbf{v}_k = \mathbf{A} \sum_{i \in D_S^k} \mathbf{e}_i$ denotes the accumulated semantic profile, while D_S^k and D_P^k represent the selected subset and the remaining candidate pool at iteration k, respectively.

To accommodate different data curation needs, we extend the framework with two sampling modes. ***General Sampling*** aims to select a high-quality and diverse subset over the entire Tag Tree, while ***Aligned Sampling*** further encourages the selected subset to match a desired knowledge distribution. Specifically, we define a target distribution $Q \in \Delta^{|V_{\text{leaf}}|}$ over the leaf nodes of the Tag Tree for aligned sampling. Since leaf nodes correspond to atomic knowledge units, Q provides fine-grained control over the expected composition of the selected data. It can be either manually specified for domain customization or estimated from a reference benchmark for benchmark-aligned selection.

To unify the two modes, we incorporate a KL-divergence-based regularization term into the greedy selection objective. Let the empirical distribution of the current selected subset be defined as $P(D_S)$. At each iteration, we select the candidate

Algorithm 1 TAGS Sampling

Input: Initial Data Pool D_P, Sample Budget N, Target Distribution Q
Output: The Sampled Dataset D_S
Initialize Empty D_S;
Initialize Propagation Matrix A;
while $|D_S| < N$ **do**
  $G \leftarrow \Phi'(A \sum_{k \in D_S} \mathbf{e}_k)^\top A$;
  $d_i \leftarrow \arg\max_{d \in D_P} G\mathbf{e}_d - \lambda \mathcal{D}_{\mathrm{KL}}(Q || P(D_S \cup \{d\}))$;
  $D_S \leftarrow D_S \cup \{d_i\}$;
  $D_P \leftarrow D_P \setminus \{d_i\}$;
end while
return D_S

sample that maximizes the following joint score:

$$d_k = \underset{d \in D_P^k}{\operatorname{argmax}} \left(\Delta\mathcal{I}(d) - \lambda \cdot \mathcal{D}_{\mathrm{KL}}(Q \parallel P(D_S^k \cup \{d\})) \right), \tag{8}$$

where $\Delta\mathcal{I}(d)$ denotes the approximated information gain defined in Eq. 7, and $\lambda \geq 0$ controls the strength of distribution alignment. When $\lambda = 0$, the objective reduces to general sampling; when $\lambda > 0$, the KL regularization encourages the selected subset to align with the target distribution Q. The detailed procedure is outlined in Algorithm 1.

4 Experiments

We evaluate the TAGS pipeline along three stages: tag extraction, hierarchical tree construction, and tree-aware data sampling. This protocol evaluates whether TAGS extracts accurate atomic concepts, builds a reliable semantic hierarchy, and improves instruction tuning under a fixed data budget.

4.1 Experimental Setup

Tagger evaluation. We first evaluate the quality of the tags produced by the tagger. To this end, we introduce TaggerEval, a human-aligned benchmark comprising 100 high-quality samples from diverse domains. Following (Lee et al., 2025), we use an LLM judge[3] to conduct tag-level binary evaluation along three dimensions: **Coverage**, whether each gold concept is covered; **Precision**, whether each generated tag faithfully reflects the input instruction; and **Fine-grained**, whether each generated tag corresponds to an atomic knowledge concept. We average scores over tags and samples, and provide the full evaluation procedure in Appendix A.1. We report the average score across these dimensions and the Cohen kappa coefficient (Sim and Wright, 2005; Zlabinger et al., 2020) between LLM and human ratings to validate judge reliability.

[3]We use `GPT-4.1` as the judge model.

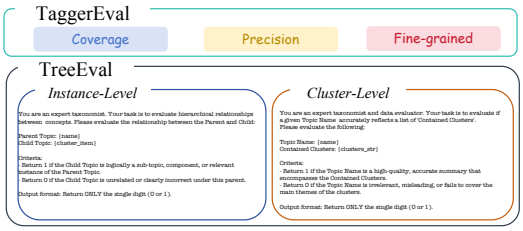

Figure 4: The TaggerEval and TreeEval frameworks.

Tree evaluation. The Tag Tree aims to organize isolated fine-grained tags into a hierarchical knowledge structure. We evaluate whether its parent-child relations are semantically coherent rather than clustering artifacts. As shown in Figure 4, TreeEval contains 200 randomly sampled tag-parent pairs and 50 parent clusters. We evaluate two aspects: **Instance-Level** correctness, which checks whether each tag is a valid subtopic of its assigned parent, and **Cluster-Level** correctness, which checks whether each parent node accurately summarizes the shared semantics of its children.

Sampling evaluation. We evaluate data selection on Tulu3 (Lambert et al., 2024) under a default 50K-sample budget, following prior work (Lu et al., 2023; Chen et al., 2025). The selected subsets are used to fine-tune Qwen3 models (Yang et al., 2025) at different scales and are evaluated with OpenCompass (Contributors, 2023). We consider two benchmark groups: knowledge-intensive tasks, including ARC (Clark et al., 2018), BBH (Suzgun et al., 2023), MMLU (Hendrycks et al., 2020), HumanEval (Chen, 2021), GSM8K (Cobbe et al., 2021), and IFEval (Zhou et al., 2023b); and open-ended human-preference tasks, including AlpacaEval2 (Dubois et al., 2024), MTBench (Zheng et al., 2023b), and WildBench (Lin et al., 2024). We report normalized percentage scores with Avg_{obj}, Avg_{sub}, and their mean Avg as aggregate metrics.

Baselines. We compare TAGS with representative data selection baselines spanning the main selection signals: random sampling (Xia et al., 2024), quality filtering (IFD (Li et al., 2024a)), tag-based signals (*#InsTag* (Lu et al., 2023)), diversity-aware signals (DEITA (Liu et al., 2024c), CaR (Ge et al., 2024), QDIT (Bukharin et al., 2024)), information gain (MIG (Chen et al., 2025)), token-level quality (T-SHIRT (Fu et al., 2026)), and pairwise utility (DELIFT (Agarwal et al., 2025)). All methods are evaluated under 50K-sample budget. Additional details are provided in Appendix A.2.

Base Model	Method	DataSize	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
Qwen3-8B	Pool	939K	88.47	69.71	88.86	77.44	75.27	**69.87**	78.27	9.69	7.34	44.52	42.54	60.40
	Random	50K	80.68	72.29	90.75	82.93	78.60	64.14	78.23	9.44	7.51	41.74	42.09	60.16
	IFD	50K	**90.85**	**77.65**	91.36	81.71	78.79	57.12	79.58	15.16	7.62	42.32	44.56	62.07
	#InsTag	50K	85.42	70.78	90.75	80.49	78.81	60.63	77.81	10.68	7.67	42.16	43.18	60.50
	DEITA	50K	60.00	71.59	**92.65**	**85.37**	78.61	60.44	74.78	14.04	7.62	44.16	44.80	59.79
	CaR	50K	79.66	73.40	91.36	82.32	78.91	56.19	76.97	13.29	7.49	42.52	43.57	60.27
	QDIT	50K	76.95	77.40	90.67	81.71	78.82	58.23	77.30	14.04	7.41	40.90	43.01	60.16
	MIG	50K	76.61	74.49	91.51	79.88	**79.04**	65.80	77.89	16.65	7.59	43.49	45.35	61.46
	T-SHIRT	50K	82.29	72.21	90.88	85.22	78.48	67.61	79.45	12.59	7.38	44.37	43.59	61.52
	DELIFT	50K	86.44	71.94	91.32	82.78	78.60	65.02	79.35	10.87	7.53	43.09	43.09	61.22
	Ours	50K	86.78	74.22	91.51	82.32	78.75	66.91	**80.08**	**20.12**	**7.85**	**44.76**	**47.79**	**63.93**
Qwen3-4B	Random	50K	85.42	67.75	89.23	78.05	75.29	61.18	76.15	9.07	7.25	38.16	39.91	58.03
	IFD	50K	91.53	72.40	78.17	76.22	75.23	57.86	75.23	**15.65**	7.26	40.15	42.80	59.02
	#InsTag	50K	84.07	66.88	86.96	78.66	75.35	63.59	75.92	9.57	7.38	38.26	40.54	58.23
	Deita	50K	76.95	67.83	81.35	78.66	**75.56**	63.22	73.93	8.20	7.26	38.14	39.64	56.78
	CaR	50K	91.86	68.51	79.30	77.44	75.26	54.90	74.54	13.54	7.33	38.80	41.88	58.21
	QDIT	50K	91.19	72.90	85.06	76.83	75.16	58.78	76.65	12.80	7.24	38.86	41.35	59.00
	MIG	50K	91.53	72.10	70.43	**81.10**	75.47	65.80	76.07	14.04	**7.64**	40.66	**43.70**	59.88
	Ours	50K	**92.54**	70.78	**89.69**	76.83	75.33	**66.73**	**78.65**	14.78	7.34	**40.98**	43.05	**60.85**
Qwen3-1.7B	Random	50K	78.64	51.68	77.26	65.85	64.33	42.14	63.32	4.84	6.64	25.02	32.09	47.70
	IFD	50K	78.31	53.92	54.97	67.07	64.20	35.49	58.99	6.21	6.60	25.46	32.56	45.77
	#InsTag	50K	78.31	50.88	77.26	66.46	64.27	43.44	63.44	5.09	6.51	26.17	32.12	47.78
	Deita	50K	71.53	51.42	62.55	65.85	64.29	42.33	59.66	5.22	6.33	25.40	31.31	45.48
	CaR	50K	71.86	52.70	73.77	61.59	**64.36**	38.45	60.45	6.46	6.43	26.42	32.40	46.42
	QDIT	50K	76.95	**54.62**	58.07	67.07	64.13	37.71	59.76	7.08	6.26	24.49	31.39	45.58
	MIG	50K	80.34	53.53	78.77	66.46	64.00	44.36	64.58	9.69	6.54	27.92	34.34	49.46
	Ours	50K	**82.03**	54.31	**80.14**	**67.68**	63.31	**48.43**	**65.98**	**10.93**	**6.76**	**29.83**	**36.12**	**51.05**

Table 1: Comparison of data selection methods on Tulu3. Avg is the mean of Avg_{obj} and Avg_{sub}.

Method	Coverage	Precision	Fine-grained	Avg.
#InsTagger	0.4527	0.2514	0.1286	0.2776
Ours (w/o Act-Critic)	0.7387	0.7815	0.6328	0.7177
Ours (w/ Act-Critic)	**0.7623**	**0.8170**	**0.7793**	**0.7862**
kappa (vs human)	0.7069	0.7826	0.7218	-

Table 2: Tagger performance on TaggerEval.

Method	Cluster-Level	Instance-Level
Hierarchical Clustering	98.00%	96.00%

Table 3: Semantic validity of the constructed Tag Tree.

4.2 Tagger Quality

The effectiveness of TAGS depends on whether semantic tags can serve as reliable units for both tree construction and data selection, as they define the granularity of the resulting hierarchy. Table 2 shows that the TAGS tagger substantially outperforms *#InsTagger*, increasing the average score from 0.2776 to 0.7862. The gains are consistent across Coverage, Precision, and Fine-grained, indicating that the generated tags are more complete, accurate, and semantically specific. These results support the use of tags produced by TAGS as reliable leaf-level units for subsequent tree construction and information-gain-based sampling. Notably, Act-Critic provides an additional improvement, increasing the average score from 0.7177 to 0.7862, with gains across all evaluation dimensions. This result suggests that iterative critique and refinement improve the overall quality of synthetic tag supervision, rather than optimizing a single tagging criterion. The kappa scores show substantial agreement with human annotations, further supporting the reliability of TaggerEval.

4.3 Tree Quality

Table 3 evaluates whether the Tag Tree provides a semantically valid hierarchy for tree-aware sampling. The hierarchy achieves 98.00% Cluster-Level correctness and 96.00% Instance-Level correctness, indicating that the induced clusters are semantically coherent and that individual instances are assigned to appropriate clusters. These results show that the tree structure preserves meaningful semantic relations among knowledge units.

4.4 Main Results

As shown in Table 1, TAGS achieves the highest overall average score across all evaluated model scales, demonstrating strong effectiveness and data efficiency. On the Qwen3-8B backbone, TAGS reaches an average score of 63.93, exceeding the strongest baseline by +2.99% and the Random baseline by +6.27%. This overall improvement is supported by gains in both objective and subjective evaluations. Compared with the strongest baseline, TAGS improves Avg_{obj} by +0.63% and achieves a larger improvement of +5.38% on Avg_{sub}. These results indicate that tree-aware

general sampling preserves the capability of the model on knowledge-intensive benchmarks while yielding stronger instruction-following quality and human-preference alignment in open-ended scenarios. More importantly, with only 50K selected samples, corresponding to approximately 5% of the full 939K data pool, the model trained on TAGS-selected data outperforms the model trained on the entire pool by +5.84%. This result suggests that the proposed sampling strategy effectively filters out redundant or noisy data and identifies a compact subset with higher training utility.

TAGS also exhibits strong scalability across model sizes. It ranks first in overall average score on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, improving over the strongest baseline by **+3.21%**, **+1.62%**, and **+2.99%**, respectively. The improvement is particularly notable on the 1.7B model, where TAGS increases both objective and subjective averages. This suggests that structured knowledge coverage and semantic diversity are especially beneficial when model capacity is limited. Overall, these results demonstrate that TAGS provides robust and scalable gains across different backbone sizes while achieving better performance with substantially fewer training samples. Additional experiments on Llama and Mistral backbones are reported in Appendix B.5, further validating the generality of TAGS across model families.

4.5 Aligned Sampling

Method	Data Size	ARC	GSM8K	GPQA-Diamond
TAGS	10K	91.19	85.67	30.30
TAGS-*Align*	10K	93.90	89.69	31.83
Improvement	-	**+2.97%**	**+4.69%**	**+5.06%**

Table 4: Performance of aligned sampling strategy.

General sampling selects a broadly useful subset, whereas aligned sampling further examines whether the Tag Tree can support controllable, target-oriented data selection. To evaluate this capability, we map ARC, GSM8K, and GPQA-Diamond (Rein et al., 2024) onto the Tag Tree and perform aligned sampling with a 10K budget and $\lambda = 5$. The target distribution is defined over leaf concepts, enabling the sampler to favor data aligned with the target domain while preserving the information-gain objective.

As shown in Table 4, TAGS-*Align* consistently outperforms general TAGS under the same data budget, achieving relative gains of 2.97%, 4.69%, and 5.06% on ARC, GSM8K, and GPQA-Diamond, respectively. These improvements span both broad knowledge evaluation and specialized reasoning benchmarks, suggesting that the alignment term provides general controllability rather than benchmark-specific tuning. The results further demonstrate that TAGS can adapt the selected data distribution through the unified tree-aware formulation, without relying on separate domain-specific filtering heuristics.

4.6 Ablation Analysis

Method	Data Size	Avg_{obj}	Avg_{sub}	Avg
MIG	50K	77.89	45.35	61.46
TAGS(w/o Tree)	50K	80.05	47.00	63.52
TAGS	50K	**80.08**	**47.79**	**63.93**

Table 5: Ablation study of fine-grained tags and tree structure.

Table 5 separates the contributions of fine-grained tagging and hierarchical organization. Replacing the coarse tags used in MIG with tags generated by the TAGS tagger improves the average score from 61.46 to 63.52, yielding the largest ablation gain. This indicates that fine-grained semantic units are the main source of improvement, as they help distinguish skills within broad domains and improve data valuation and subset coverage. Building the Tag Tree further raises the score to 63.93, especially on subjective evaluation, confirming the benefit of organizing semantic units across multiple levels rather than treating them as flat labels. This hierarchy helps the sampler estimate redundancy and coverage more accurately, enabling TAGS to select examples with new structured information instead of repeatedly choosing samples with similar leaf-level concepts. Additional analyses of data scale and hyperparameter sensitivity are provided in Appendix B.

5 Conclusion

In this paper, we present TAGS, a framework that integrates fine-grained tags into a hierarchical tree for controllable data selection. By organizing atomic tags into a global taxonomy, TAGS enables sampling that balances data quality and diversity. Extensive experiments show that TAGS consistently outperforms strong baselines across diverse benchmarks, achieving superior performance with only 5% of the original data, demonstrating its efficacy for efficient instruction tuning.

Limitations

1. **Tagger Data Scale:** To balance performance with proprietary API costs, our Tagger was fine-tuned on a compact dataset of 53K samples. While this effectively validates data efficiency, it precludes a comprehensive analysis of the Tagger's scaling behavior on larger synthetic datasets.

2. **Model & Task Scope:** Due to computational constraints, our evaluation focuses on the `Qwen3-1.7B`, `4B`, `8B` series for SFT tasks. Verification on larger-scale models (e.g., `Qwen3-30B`) and other paradigms like Reinforcement Learning remains for future exploration.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ishika Agarwal, Krishnateja Killamsetty, Lucian Popa, and Marina Danilevsky. 2025. Delift: Data efficient language model instruction fine-tuning. In *International Conference on Learning Representations*, volume 2025, pages 100488–100509.

Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. 2024. Data diversity matters for robust instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3411–3425.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and 1 others. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yicheng Chen, Yining Li, Kai Hu, Ma Zerun, HaochenYe HaochenYe, and Kai Chen. 2025. Mig: Automatic data selection for instruction tuning by maximizing information gain in semantic space. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9902–9915.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

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

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. 2024. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*.

Yanjun Fu, Faisal Hamman, and Sanghamitra Dutta. 2026. T-shirt: Token-selective hierarchical data selection for instruction tuning. *Advances in Neural Information Processing Systems*, 38:113932–113958.

Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Mahong Xia, Zhang Li, Boxing Chen, Hao Yang, and 1 others. 2024. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 464–478.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

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

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Yukyung Lee, Joonghoon Kim, Jaehee Kim, Hyowon Cho, Jaewook Kang, Pilsung Kang, and Najoung Kim. 2025. Checkeval: A reliable llm-as-a-judge framework for evaluating text generation using checklists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15782–15809.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Jijie Li, Li Du, Hanyu Zhao, Bo-wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. 2025. Infinity instruct: Scaling instruction selection and synthesis to enhance language models. *arXiv preprint arXiv:2506.11116*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024a. 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)*, pages 7602–7635.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. 2023. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024b. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *ACL*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liangxin Liu, Xuebo Liu, Derek F Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024b. Selectit: Selective instruction tuning for llms via uncertainty-aware self-reflection. *Advances in Neural Information Processing Systems*, 37:97800–97825.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024c. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *arXiv preprint arXiv:2308.07074*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, and 1 others. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Julius Sim and Chris C Wright. 2005. The kappa statistic in reliability studies: use, interpretation, and sample size requirements. *Physical therapy*, 85(3):257–268.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, and 1 others. 2024. Clio: Privacy-preserving insights into real-world ai use. *arXiv preprint arXiv:2412.13678*.

M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Kang Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixing Deng, Shuyue Guo, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, and 76 others. 2025. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *Preprint*, arXiv:2502.14739.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Hemish Veeraboina. 2023. Aime problem set 1983-2024.

David Wadden, Kejian Shi, Jacob Morrison, Alan Li, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, Shannon Zejiang Shen, Doug Downey, Hannaneh Hajishirzi, and Arman Cohan. 2025. SciRIFF: A resource to enhance language model instruction-following over scientific literature. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6083–6120, Suzhou, China. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Tingyu Xia, Bowen Yu, Kai Dang, An Yang, Yuan Wu, Yuan Tian, Yi Chang, and Junyang Lin. 2024. Rethinking data selection at scale: Random selection is almost all you need. *arXiv preprint arXiv:2410.09335*.

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. 2025. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms. *arXiv preprint arXiv:2504.14655*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. 2025. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6980–7008, Vienna, Austria. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Mingjia Yin, Chuhan Wu, Yufei Wang, Hao Wang, Wei Guo, Yasheng Wang, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. 2024. Entropy law: The story behind data compression and llm performance. *arXiv preprint arXiv:2407.06645*.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Dong Wang, Ilia Kulikov, Kyunghyun Cho, Yuandong Tian, Jason E Weston, and Xian Li. 2025. Naturalreasoning: Reasoning in the wild with 2.8m challenging questions. *Preprint*, arXiv:2502.13124.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2023a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *Preprint*, arXiv:2309.11998.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023a. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.

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

Markus Zlabinger, Marta Sabou, Sebastian Hofstätter, Mete Sertkan, and Allan Hanbury. 2020. Dexa: Supporting non-expert annotators with dynamic examples from experts. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 2109–2112, New York, NY, USA. Association for Computing Machinery.

A More TAGS Implementation Details

A.1 Details of TaggerEval

For each sample x_i, let $\mathcal{G}_i = \{g_{i1}, \ldots, g_{iM_i}\}$ denote the set of gold tags, and let $\mathcal{T}_i = \{t_{i1}, \ldots, t_{iN_i}\}$ denote the set of generated tags. We evaluate the generated tags from three perspectives: **Coverage**, **Precision**, and **Fine-grained**. Specifically, coverage measures the proportion of gold tags captured by the generated tag set; precision measures the proportion of generated tags that match the gold tag set; and fine-grained evaluates whether each generated tag corresponds to an atomic knowledge concept.

For coverage, the judge checks whether each gold tag g_{im} is semantically covered by any generated tag in $\mathcal{T}_i$. The coverage score is defined as:

$$\text{Coverage}(x_i) = \frac{1}{M_i} \sum_{m=1}^{M_i} \mathbb{I}(g_{im} \text{ is covered by } \mathcal{T}_i).$$

For precision, the judge checks whether each generated tag t_{in} is semantically matched by any

Domain	Quantity	Source
General	10K	Magpie-Pro (Xu et al., 2024), Infinity-Instruct (Li et al., 2025), OpenHermes (Teknium, 2023), ...
Math	10K	NuminaMath-1.5 (LI et al., 2024), DeepMath-103K (He et al., 2025), Aqua-Rat (Ling et al., 2017), ...
Code	10K	TACO (Li et al., 2023), KODCode (Xu et al., 2025), LeetCodeDataset (Xia et al., 2025), ...
STEM	10K	Natural-Reasoning (Yuan et al., 2025), MegaScience (Xu et al., 2024), SciRiFF (Wadden et al., 2025), ...
Chat	10K	lmsys-Chat (Zheng et al., 2023a)
Complex	3k	AIME (Veeraboina, 2023), ArenaHard(Math, Coding) (Li et al., 2024b), SuperGPQA (Team et al., 2025), ...

Table 6: Domain distribution and source datasets used in tagger training.

gold tag in $\mathcal{G}_i$. The precision score is defined as:

$$\text{Precision}(x_i) = \frac{1}{N_i} \sum_{n=1}^{N_i} \mathbb{I}(t_{in} \text{ matches } \mathcal{G}_i).$$

For fine-grained evaluation, the judge evaluates whether each generated tag t_{in} represents an atomic knowledge concept. The fine-grained score is defined as:

$$\text{Fine-grained}(x_i) = \frac{1}{N_i} \sum_{n=1}^{N_i} \mathbb{I}(t_{in} \text{ is atomic}).$$

The final score for each dimension is obtained by averaging the corresponding sample-level scores over all evaluated samples.

A.2 Baseline Settings

We follow the comparison protocol established by MIG (Chen et al., 2025) and evaluate all methods under a fixed 50K-sample SFT budget. For MIG-released baselines, we directly use their sampled subsets[4]. For additional recent baselines, including DELIFT (Agarwal et al., 2025) and T-SHIRT (Fu et al., 2026), we rerun their selection procedures on the same candidate pool and retain 50K samples for fair comparison. All subsets are fine-tuned and evaluated under identical settings, ensuring that performance differences mainly reflect the intrinsic effectiveness of each data selection method.

A.3 Implementation Details and Efficiency Analysis

A.3.1 Tagger Training.

We first collect 53k Supervised Fine-Tuning samples synthesized by `GPT-o4-mini` using an Actor–Critic pipeline to train the TAGS Tagger, and then further fine-tune it on `Qwen3-8B-Instruct`[5]. The distribution of the training data is summarized in Table 6. For non-complex queries, we apply uniform sampling over the entire pool. For complex queries, we perform uniform sampling from several inherently high-difficulty datasets spanning the Math, Code, and STEM domains:

- **Math**: AIME and ArenaHard-Math.
- **Code**: ArenaHard-Coding.
- **STEM**: SuperGPQA.

A.3.2 Tree Construction.

We adopt the bottom-up hierarchical clustering algorithm described in Clio (Tamkin et al., 2024). Starting from 7M high-quality open-source samples (Teknium, 2023; Xu et al., 2024; LI et al., 2024; Xu et al., 2025; Li et al., 2025), we first apply the TAGS Tagger to extract fine-grained tags. To ensure computational efficiency at scale, we use all-mpnet-base-v2[6] as the encoder for initial coarse-grained clustering, reducing the data into 10,000 preliminary clusters. We then employ `Qwen3-30B-A3B-2507`[7] to summarize, deduplicate, reassign, and rename cluster labels, producing cleaner and more semantically coherent tag groups. After this refinement step, we further apply Qwen3-Embedding-0.6B[8] for subsequent clustering, which provides a better balance between clustering quality and computational cost. This multi-stage pipeline ultimately yields a 10-level hierarchical tree, offering a structured and interpretable organization of the underlying data space.

Importantly, the tag tree is constructed as a one-time offline preprocessing step and can be reused across different downstream selection runs and target models. Therefore, although tree construction introduces an upfront cost, this cost is amortized over repeated use.

[4] https://huggingface.co/collections/xsample/mig-datasets

[5] https://huggingface.co/Qwen/Qwen3-8B

[6] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[7] https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507

[8] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

Stage	Time (min)
General Sampling	3.13
Aligned Sampling	35.51
SFT (50k Data)	90.85
SFT (939k Data)	1810.53

Table 7: Computational overhead of sampling and training phases.

A.3.3 Sampling and Training Setup.

We sample 50k training instances from the full 939k Tulu3 dataset (Lambert et al., 2024). Both the sampling process and subsequent SFT training are accelerated with 8 GPUs. In the initial stage of sampling, we initialize G as a zero vector. For training, we set the batch size to 128, learning rate to 5e-6, warmup ratio to 0.03, and maximum sequence length to 8192. The implementation is conducted using the Swift framework (Zhao et al., 2024).

As shown in Table 7, General Sampling is highly efficient, taking only 3.13 minutes and occupying less than 0.2% of the full-data SFT time. Even Aligned Sampling, which involves additional KL-divergence calculations, takes only 35.51 minutes, indicating that the computational overhead remains limited. Moreover, training on the selected 50k subset reduces the SFT time from 1810.53 minutes to 90.85 minutes, around 5% of the full-data training cost, while achieving superior performance.

A.3.4 Computational Efficiency and Reusability.

We further analyze the computational cost of TAGS. In practice, TAGS is designed to avoid repeatedly applying expensive LLM operations to all raw samples. For a collection of M samples, we first apply a fine-tuned Qwen3-8B model as the TAGS Tagger, resulting in one tag extraction call for each sample. After this one-time tagging pass, tree construction is performed over the induced tag space, and LLM-based refinement is mainly applied to cluster-level labels rather than every raw sample. This design yields a reusable data-centric structure for downstream selection and substantially reduces repeated model-dependent computation.

We then analyze the cost of tree construction. Let N denote the number of leaf nodes in the Tag Tree and L denote the tree depth. The number of LLM calls required for tree construction is

$$\begin{aligned} T_{\text{tree}} &= N + \sum_{k=1}^{L-1} \left(3N\lambda^k + N\lambda^{k-1}\right) \\ &= N + (N-1)\frac{1+3\lambda}{1-\lambda}, \end{aligned} \tag{9}$$

where $\lambda = (\frac{1}{N})^{\frac{1}{L-1}}$ controls the layer-wise clustering ratio. Since $N - 1 \leq N$, we have

$$T_{\text{tree}} \leq N + N\frac{1+3\lambda}{1-\lambda}. \tag{10}$$

When $\lambda \leq 1/2$, equivalently $N \geq 2^{L-1}$, the tree construction cost satisfies

$$T_{\text{tree}} \leq 6N. \tag{11}$$

Thus, aside from the one-time tagging pass, TAGS incurs only linear tree construction cost in the number of leaf nodes, rather than repeatedly scaling with the number of raw samples.

A.3.5 Incremental Tree Update.

Although the tag tree is constructed offline, it is not restricted to a fixed data distribution. To handle newly emerging tags, domains, or instruction types, TAGS supports localized incremental updates without rebuilding the entire tree. Given a new tag l, we perform bottom-up semantic matching along the existing hierarchy. At each level, if an existing parent node m semantically subsumes l, the new tag is inserted as a child of m. Otherwise, a new node is created, and the matching process continues upward until a suitable position is found. This update mechanism only affects a local path and a small number of neighboring nodes in the tree. Therefore, it is substantially cheaper than full reconstruction and makes the tag tree more practical under evolving data distributions or concept drift.

B Detailed Experimental Results

We provide additional experimental results to complement the main analysis. Specifically, we report detailed results under different data scales, hyperparameter settings, fine-grained ablation results on each benchmark, compute-controlled comparison, cross-model generalization, and the effect of tree depth. Among them, Figure 5 visualizes the overall trends of data size, α, and γ, while the remaining experiments are reported in the corresponding tables. Together, these results offer a more comprehensive view of the effectiveness, robustness, and practical utility of TAGS.

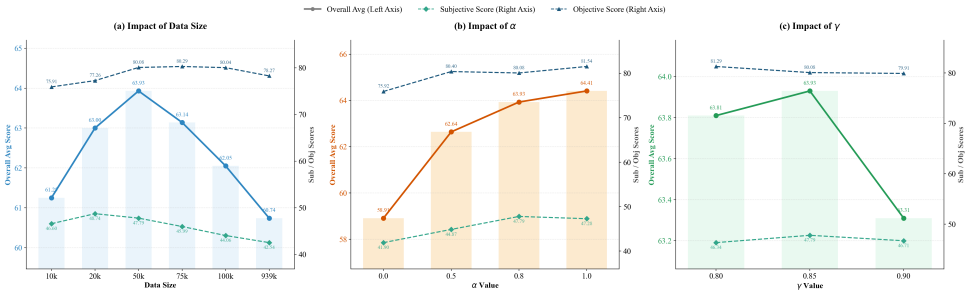

Figure 5: Ablation studies on training data size, hyperparameter α of complicated score s, and γ value of $\Phi(x)$.

Method	DataSize	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
TAGS	10k	**91.19**	70.15	85.67	69.51	78.49	60.44	75.91	18.63	7.74	43.77	46.60	61.25
	20k	85.42	70.16	86.88	79.27	76.74	65.06	77.26	**22.11**	**7.85**	**45.62**	**48.74**	63.00
	50k	86.78	**74.22**	91.51	**82.32**	78.75	66.91	80.08	20.12	**7.85**	44.76	47.79	**63.93**
	75k	86.78	74.00	91.51	**82.32**	78.74	68.39	**80.29**	17.02	7.68	44.16	45.99	63.14
	100k	84.07	73.27	**91.66**	81.10	**79.01**	**71.16**	80.04	15.78	7.38	42.58	44.06	62.05
ALL	939k	88.47	69.71	88.86	77.44	75.27	69.87	78.27	9.69	7.34	44.52	42.54	60.74

Table 8: Performance comparison of different data size on objective and subjective benchmarks.

B.1 Effect of Data Scale

Table 8 reports the performance of TAGS under different data budgets. We vary the selected subset size from 10k to 100k and also include the full 939k Tulu3 dataset for comparison. The results show that increasing the data size does not monotonically improve performance. Although larger subsets provide broader coverage, they may also introduce redundant or low-utility samples, weakening the advantage of data selection.

Among all tested budgets, the 50k setting achieves the best overall average score. It provides a strong balance between objective and subjective benchmarks, outperforming both smaller subsets and the full-data setting. Notably, training on all 939k samples does not lead to the best performance, suggesting that data quality and structural coverage are more important than simply increasing the amount of training data.

B.2 Hyperparameter Sensitivity

We further analyze the sensitivity of TAGS to key hyperparameters. The hyperparameter α controls the balance between quality and complexity in the sample utility score, while γ controls the strength of the tree-aware information gain during sampling. For α, Table 9 shows that relying solely on either complexity or quality is suboptimal. While setting $\alpha = 1.0$ improves average scores, incorporating complexity ($\alpha = 0.8$) yields the best results on challenging reasoning tasks (BBH, MMLU) and open-ended benchmarks (MT-Bench, WildBench), suggesting that a 20% complexity weight helps select harder, more informative samples. For γ, Table 10 shows that TAGS remains stable within a reasonable range. The setting $\gamma = 0.85$ balances noise filtering and semantic coverage, indicating that the tree-aware selection objective is not overly sensitive to nearby hyperparameter configurations.

B.3 Fine-Grained Ablation Results

Table 11 provides the benchmark-level ablation results corresponding to the ablation analysis in the main text. Here, we report the detailed scores on each objective and subjective benchmark to show how the fine-grained Tagger and Tag Tree affect different evaluation dimensions.

B.4 Compute-Controlled Comparison

To verify that the advantage of TAGS does not simply come from reducing the number of training steps or avoiding overfitting on the full dataset, we conduct a compute-controlled comparison. Specifically, we train a model on the full data pool using the same number of optimization steps as the TAGS-50K model. This setting controls the training compute and allows us to examine whether the selected data itself provides higher utility.

As shown in Table 12, TAGS substantially outperforms the same-step full-pool baseline. The gap is consistent across objective and subjective benchmarks, indicating that the performance gain of TAGS mainly comes from selecting higher-quality and more informative data, rather than merely reducing training time.

Method	DataSize	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
TAGS (α=0.0)	50K	80.34	73.82	78.24	84.15	78.89	60.07	75.92	10.93	7.28	41.96	41.90	58.91
TAGS (α=0.5)	50K	85.08	73.59	91.13	**84.76**	**78.89**	68.95	80.40	16.02	7.53	43.29	44.87	62.64
TAGS (α=0.8)	50K	86.78	**74.22**	91.51	82.32	78.75	66.91	80.08	20.12	**7.85**	**44.76**	**47.79**	63.93
TAGS (α=1.0)	50K	**87.80**	74.12	**92.19**	84.15	78.71	**72.27**	**81.54**	**20.75**	7.67	44.40	47.28	**64.41**

Table 9: Performance comparison of different hyperparameter α on objective and subjective benchmarks.

Method	DataSize	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
TAGS ($\gamma = 0.80$)	50K	**89.83**	73.86	**92.42**	**82.93**	**79.02**	**69.69**	**81.29**	19.13	7.55	44.38	46.34	63.81
TAGS ($\gamma = 0.85$)	50K	86.78	74.22	91.51	82.32	78.75	66.91	80.08	**20.12**	**7.85**	44.76	**47.79**	**63.93**
TAGS ($\gamma = 0.90$)	50K	86.44	**75.02**	91.05	78.66	78.79	69.50	79.91	19.38	7.59	**44.86**	46.71	63.31

Table 10: Performance comparison of different hyperparameter γ on objective and subjective benchmarks.

B.5 Cross-Model Generalization

Since TAGS is a model-agnostic data selection method, we further evaluate whether the selected subset can transfer across different backbone models. Although the tag tree is constructed using Qwen-based components, the selected data should ideally remain useful for other model architectures. To verify this, we fine-tune different backbone models using the 50K subset selected by TAGS and compare it with representative baselines.

As shown in Table 13, TAGS achieves strong overall performance across different backbone models. These results demonstrate that the selected data has transferable utility and is not tightly coupled to the model used during tree construction.

B.6 Effect of Tree Depth

We also study the effect of tree depth on the final selection quality. The tree depth controls the granularity of the hierarchical knowledge organization. A shallow tree provides a coarser semantic structure, while a deeper tree captures more fine-grained semantic relations among tags.

As shown in Table 14, deeper hierarchical structures generally improve subjective evaluation and overall performance. This suggests that fine-grained hierarchical organization is useful for preserving semantic diversity and knowledge coverage, especially for open-ended generation. Meanwhile, the performance gains gradually diminish as depth increases, indicating that the chosen tree depth provides a reasonable balance between effectiveness and computational cost.

C Prompt Template

C.1 Tagging Data Synthesis

We instruct an LLM to act as GPT-Tagger for initial tagging (Table 15), then act as a GPT-Checker for tag refinement (Table 16) to synthesize high-quality tagger training data.

C.2 TaggerEval

We prompt an LLM as a judge to evaluate the generated tags from coverage (Table 17), precision (Table 18), fine-grained (Table 19) dimensions.

C.3 TreeEval

We further prompt the LLM to evaluate the constructed Tag Tree from Instance-Level (Table 20) and Cluster-Level (Table 21) dimensions.

Method	DataSize	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
MIG	50K	76.61	**74.49**	91.51	79.88	**79.04**	65.80	77.89	16.65	7.59	43.49	45.35	61.46
TAGS(w/o Tree)	50K	85.08	74.05	**91.58**	**83.54**	78.98	**67.10**	80.05	**20.25**	7.58	**44.95**	47.00	63.52
TAGS	50K	**86.78**	**74.22**	91.51	82.32	78.75	66.91	**80.08**	20.12	**7.85**	44.76	**47.79**	**63.93**

Table 11: Detailed performance of ablation study on objective and subjective benchmarks.

Layer	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
Same Steps	69.28	73.28	87.75	71.34	78.54	58.55	73.12	6.55	7.06	36.37	37.84	55.48
Ours	**86.78**	**74.22**	**91.51**	**82.32**	**78.75**	**66.91**	**80.08**	**20.12**	**7.85**	**44.76**	**47.79**	**63.93**

Table 12: Comparison under matched compute budget.

Method	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
Mistral-7B-v0.1												
Random	75.97	57.42	**66.40**	39.71	61.31	65.06	60.98	5.79	7.06	36.21	37.53	49.26
Instag	78.77	**57.81**	63.56	44.21	**62.99**	65.80	62.19	13.12	7.07	39.64	41.15	51.67
MIG	77.29	57.15	64.53	**44.74**	61.57	65.59	61.81	8.01	6.92	37.60	38.27	50.04
Ours	**80.34**	56.85	65.96	43.29	61.67	**66.36**	**62.41**	**15.05**	**7.20**	**39.94**	**42.33**	**52.37**
LLaMA-3-8B												
Random	**70.85**	**65.98**	66.72	51.83	65.06	57.49	**62.98**	3.92	6.71	30.19	33.74	48.36
Instag	63.52	64.92	**69.95**	48.70	**65.19**	58.41	61.78	4.19	6.57	30.25	33.38	47.58
MIG	64.11	64.58	64.59	47.71	64.74	58.55	60.71	8.85	6.75	34.12	36.82	48.76
Ours	67.12	63.83	66.26	**55.49**	64.18	**59.52**	62.73	**12.38**	**6.87**	**35.35**	**38.81**	**50.77**

Table 13: Cross-model generalization results under the 50K data budget.

Layer	ARC	BBH	GSM	HE	MMLU	IFEval	Avg_{obj}	AE	MT	Wild	Avg_{sub}	Avg
L=3	**87.80**	75.81	91.74	**84.15**	78.70	65.25	**80.57**	17.76	7.67	42.98	45.81	63.19
L=5	87.12	**75.91**	91.89	82.02	78.95	65.43	80.22	19.50	7.78	43.44	46.91	63.57
L=7	85.76	75.49	**92.12**	82.93	**79.21**	65.99	80.25	19.50	**7.88**	43.69	47.33	63.79
L=10	86.78	74.22	91.51	82.32	78.75	**66.91**	80.08	**20.12**	7.85	**44.76**	**47.79**	**63.93**

Table 14: Effect of tree depth on performance (50K data).

Instruction template for GPT-Tagger

You are a general knowledge extraction expert. Given a problem and its corresponding solution (which can be code, natural language, or other forms), your task is to identify all the fine-grained and atomic-level knowledge components involved.

Improvement Guidance
- Previous Tag: {old_tags}
- Improvement Hint: {improvement_reason} - If "None": This is a cold start, generate tags normally.
- If not "None": This is feedback from the checker, refine your output accordingly.
- For tags without explicitly identified issues, please keep them as they are. Only adjust the parts where problems are indicated.

You must consider multiple dimensions, including but not limited to:
* **Problem Domain** (e.g., "Number Theory", "Greedy Algorithms", "Ancient Rome History"),
* **Problem Type** (e.g., "Calculation", "Proof", "Code Implementation", "Conceptual Explanation", "Dialogue Response"),
* **Knowledge / Skill Points** (e.g., "Greatest Common Divisor", "Prime Factorization", "Binary Search", "Close Reading"),
* **User Intent** (e.g., "Learning a concept", "Practical application", "Exam preparation"),

Each knowledge point must be:
* **Atomic**: the smallest meaningful concept or technique involved.
* **Precise**: no vague categories like "Math" or "Data Structures".
* **Unabbreviated**: e.g., use "Dynamic Programming" instead of "DP".
* **Non-overlapping**: do not include multiple tags that describe the same idea with slight variation; only one representative tag should be included.
* **Core-only**: include only the most essential knowledge points necessary to understand or implement the solution. Do not include secondary or peripheral techniques unless they are critical.
* **Specifically Naming**: Your tag should be specific, include the core concept and entities into the tag.
* **Focus** on the user's primary intent while ignoring irrelevant context.
* If the problem is in a multi-turn dialogue format, use the previous conversation history as contextual background, but focus primarily on the current query.
* **Maximally concise**: output no more than *5* knowledge points.

Output Format
Return your results as a list of dictionaries, where each dictionary includes
* ''tag'': the name of the knowledge point
* ''explanation'': a brief and accurate explanation of its role in the problem and solution

Example
Problem: Find the GCD of 84 and 60
Reference Answer: Let's find the **GCD (Greatest Common Divisor)** of 84 and 60. ### Step 1: Prime factorization
* $84 = 2 \times 2 \times 3 \times 7 = 2^2 \times 3^1 \times 7^1$
$60 = 2 \times 2 \times 3 \times 5 = 2^2 \times 3^1 \times 5^1$
Step 2: Take the lowest powers of common primes
Common primes: 2 and 3
* $2^{\min(2,2)} = 2^2 = 4$
$3^{\min(1,1)} = 3^1 = 3$
Now multiply:

$$4 \times 3 = 12$$

GCD(84, 60) = 12
Output:

```
[{"tag": "Greatest Common Divisor", "explanation": "The problem requires computing the largest integer that divides 84 and 60."},
 {"tag": "Prime Factorization", "explanation": "One method of finding the GCD involves decomposing numbers into prime factors."},
 {"tag": "Euclidean Algorithm", "explanation": "An efficient iterative method to compute the GCD without full factorization."}]
```

Problem: Where is the Peking University?
Reference Answer: No. 5 Yiheyuan Road, Haidian District, Beijing, China.
Output:

```
[{"tag": "Peking University Location", "explanation": "Identifying the geographical location of Peking University,
including its city and district."},
{"tag": "Beijing Geography", "explanation": "Knowledge of the districts of Beijing to contextualize the university's location."},
{"tag": "Higher Education Institutions in China", "explanation": "Understanding the placement and distribution of
major universities in China for reference."}]
```

Problem:Which Sun sign, Rising sign, and Moon sign does a person born on September 17 (Gregorian calendar) belong to? Why are there differences among them?
Reference Answer:Sun sign: Virgo (because September 17 falls within the Virgo date range). Rising sign (Ascendant): Requires the **exact time and place of birth** to determine. Moon sign: Also requires the **exact time and place of birth** to calculate. The Sun sign depends only on the birth date, while the Rising sign and Moon sign vary with the **specific birth time and location**, as they change much more rapidly.
Output:

```
[{"tag": "Sun Sign Determination", "explanation": "Identifying the sun sign based on the Gregorian date of birth"},
{"tag": "Ascendant Sign Calculation", "explanation": "Determining the rising sign based on precise birth time and location"},
{"tag": "Moon Sign Calculation", "explanation": "Determining the moon sign based on the moon's position at the time of birth"},
{"tag": "Astrological Principles", "explanation": "Understanding why sun sign, moon sign, and ascendant can
differ due to celestial mechanics and astrological rules"}]
```

**Now analyze the following input and return the list of atomic knowledge points. You should only output the final json. **

Problem:
{problem}

Reference Answer:
{reference_answer}

Output: '''

Table 15: Instruction template for GPT-Tagger Eval.

Instruction template for GPT-Checker

You are a tag validation expert.
You are given a problem description, reference answer and the extracted tag results.
Your task is to **evaluate the tag results** based on the following criteria:

Validation Criteria
1. **Output Format**:
- Must be a JSON object.
- "tags" must be a list of dictionaries, each with fields "explanation" and "tag".

2. **Content Reasonableness**:
- Tags must match the problem description.
- Tags must be meaningful and reasonably specific.

3. **Accuracy and Granularity**:
- No vague categories (e.g. "Math" or "Data Structures").
- Tags should not be overly broad (e.g. "Probability") nor overly fine-grained (e.g. "apply addition in Bayes formula").
- Aim for an appropriate middle-level granularity (e.g. "Discrete Event Probability", "Binary Search Tree Traversal", "Video file handling in Python").
- Tags must be correct, contains the core constraints and relevant to the problem.

4. **Coverage**:
- Tags should cover the **essential core knowledge points** required to understand or solve the problem.
- Maximum 5 tags; unnecessary extra tags should not appear.

Your Task
- If the output fully satisfies the criteria, respond with:
"check":"Yes"

- If the output fails in any aspect, respond with:
"check":"No", "Reason": "<concise reason for failure, with direct guidance to refine the tagging>"

—
Problem:
{problem_description}

Reference Answer:
{reference_answer}

Extracted Tags:
{tag_output}

Table 16: Instruction template for GPT-checker.

Instruction template for Coverager Eval

You are given two tag lists:
1. **Raw Tags**: These are the ground-truth tags that represent concepts or labels in the data.
2. **Generated Tags**: These are tags automatically generated by a system.

Your task is to check whether each raw tag is **covered** by the generated tags. Please determine whether each raw tag was covered by the generated tag.

Coverage criteria include:
1. Synonymous or closely related expressions If the generated tag expresses the same or very similar meaning as the raw tag (including synonyms or paraphrases).
2. Containment relationship If the generated tag directly contains the core concept of the raw tag, it counts as coverage.For example, "subarray counting" covers "array".
3. Hypernym-hyponym coverage If the generated tag is a more specific concept or includes the core meaning of the raw tag, it is considered coverage. For example, "sliding window algorithm" covers "array".
4. Consistent domain terminology Different expressions of the same technical term count as mutual coverage. For example, "greatest common divisor" and "GCD" cover each other.
5. Composite tag inclusion If the generated tag is a composite concept and its core includes the raw tag's core term with precise semantics, it counts as coverage. For example, "quick sort algorithm" covers "sorting".

Excluded cases (do not count as coverage):
Overly broad or generic tags are not considered coverage, even if they belong to the same general domain.
The generated tag must express a specific concept that directly includes or aligns with the raw tag's meaning.

Output format:
Return a dictionary, where each key is a raw tag, and the value is '1' if it is covered by any generated tag, otherwise '0'.

Example:
Raw Tags: ["Named Entity Recognition", "POS Tagging", "DFS"]
Generated Tags: ["Part-of-Speech Tagging", "NER"]
Output: {{ "Named Entity Recognition": 1, "POS Tagging": 1, "DFS": 0 }}

You should output the dict in str format only, no analysis. Please carefully consider the intrinsic relationship between the generated tag and the raw tag, not just based on surface-level meaning. You may also make reasonable inferences — for example, character-level operations can be considered part of the string category.

Now evaluate:
Raw Tags: {raw_tags}
Generated Tags: {generate_tags}

Table 17: Instruction template for Coverage Eval.

Instruction template for Precision Eval

You are given two tag lists:
1. **Raw Tags**: These are the ground-truth tags that represent concepts or labels in the data.
2. **Generated Tags**: These are tags automatically generated by a system.

Your task is to check whether each generated tag is **accurate**, meaning it correctly corresponds to at least one raw tag.

Accuracy criteria include:
1. Synonymous or closely related expressions
If the generated tag expresses the same or very similar meaning as any raw tag (including synonyms or paraphrases).
2. Containment relationship
If the generated tag directly contains the core concept of any raw tag, it counts as accurate. For example, "subarray counting" is accurate if the raw tags contain "array".
3. Hypernym-hyponym relationship
If the generated tag is a more specific concept or includes the core meaning of a raw tag, it is considered accurate.
4. Consistent domain terminology
Different expressions of the same technical term count as mutual accuracy. For example, "greatest common divisor" and "GCD" are accurate matches.
5. Composite tag inclusion
If the generated tag is a composite concept and its core includes the raw tag's core term with precise semantics, it counts as accurate.

Excluded cases (do not count as accurate):
- Overly broad or generic generated tags that do not specifically match any raw tag.
- Tags unrelated to any raw tag in meaning.

Output format:
Return a dictionary, where each key is a generated tag, and the value is '1' if it is accurate, otherwise '0'.

Example:
Raw Tags: ["Named Entity Recognition", "POS Tagging", "DFS"]
Generated Tags: ["Part-of-Speech Tagging", "NER", "Graph Algorithm"]

Output:
{{ "Part-of-Speech Tagging": 1, "NER": 1, "Graph Algorithm": 0 }}

You should only output the dict in str format, donnot output analysis.

Now evaluate:
Raw Tags: {raw_tags}
Generated Tags: {generate_tags}

Table 18: Instruction template for Precision Eval.

Instruction template for Fine-grained Eval

You are given a list of generated tags.

Your task is to determine for each tag whether it is a **fine-grained** knowledge concept.

Criteria:
- Fine-grained tags are specific, well-defined concepts or techniques.
- Tags that are broad should be marked as not fine-grained.
- A moderate level of granularity is also acceptable — for example, 'solving quadratic equations' is considered fine-grained.
- For example, "Quick Sort" is fine-grained, but "Sorting Algorithms" is not.

Output format:
Return a dictionary where each key is a generated tag, and the value is '1' if it is fine-grained, or '0' if not.

Example:
Generated Tags: ["Quick Sort", "Sorting Algorithms", "Graph Traversal"]

Output:
{{ "Quick Sort": 1, "Sorting Algorithms": 0, "Graph Traversal": 0 }}

Only output the dictionary in string format without any explanation.

Now evaluate:
Generated Tags: {generate_tags}

Table 19: Instruction template for Fine-grained Eval.

Instruction template for Instance-level Eval

You are an expert taxonomist. Your task is to evaluate hierarchical relationships between concepts. Please evaluate the relationship between the Parent and Child:

Parent Topic: {name}
Child Topic: {cluster_item}

Criteria:
- Return 1 if the Child Topic is logically a sub-topic, component, or relevant instance of the Parent Topic.
- Return 0 if the Child Topic is unrelated or clearly incorrect under this parent.

Output format: Return ONLY the single digit (0 or 1).

Table 20: Instruction template for Instance-Level Eval.

Instruction template for Cluster-Level Eval

You are an expert taxonomist and data evaluator. Your task is to evaluate if a given Topic Name accurately reflects a list of 'Contained Clusters'.
Please evaluate the following:

Topic Name: {name}
Contained Clusters: {clusters_str}

Criteria:
- Return 1 if the Topic Name is a high-quality, accurate summary that encompasses the Contained Clusters.
- Return 0 if the Topic Name is irrelevant, misleading, or fails to cover the main themes of the clusters.

Output format: Return ONLY the single digit (0 or 1).

Table 21: Instruction template for Cluster-Level Eval.