Task–Specificity Score: Measuring How Much Instructions Really Matter for Supervision

Anonymous ACL submission

Abstract

Instruction tuning is now the default way to train and adapt large language models, but many instruction–input–output pairs are only weakly specified: for a given input, the same output can remain plausible under several alternative instructions. This raises a simple question: *does the instruction uniquely determine the target output?*

We propose the **Task–Specificity Score (TSS)** to quantify how much an instruction matters for predicting its output, by contrasting the true instruction against plausible alternatives for the same input. We further introduce **TSS++**, which uses hard alternatives and a small quality term to mitigate easy-negative effects. Across three instruction datasets (ALPACA, DOLLY-15K, NI-20) and three open LLMs (Gemma, Llama, Qwen), we show that selecting task-specific examples improves downstream performance under tight token budgets and complements quality-based filters such as perplexity and IFD (Instruction Following Difficulty).

1 Introduction

Instruction tuning has become a central paradigm for adapting pretrained language models to follow natural-language instructions across a wide range of tasks (Ouyang et al., 2022; Wang et al., 2023; Taori et al., 2023). By fine-tuning on large mixtures of instruction–input–output examples, models acquire broad task generality and improved alignment with user intent. As instruction datasets continue to grow in size and diversity, however, a fundamental question remains underexplored: *which instruction examples actually provide task-defining supervision, and which are largely redundant?*

Most existing approaches to instruction data selection and weighting focus on *output quality* or *model fit*. Perplexity-based filtering prefers examples with high likelihood under a pretrained model, implicitly favoring fluent, in-distribution responses (Zhou et al., 2023; Li et al., 2024a). Instruction-following difficulty (IFD) variants compare likelihood with and without the instruction to estimate whether the instruction helps predict the output (Li et al., 2024b). More recent work studies data selection through gradient similarity, influence functions, or representational diversity (Pruthi et al., 2020; Xia et al., 2024; Zhang et al., 2025; Yeh et al., 2022). While effective in many settings, these signals answer a different question than the one we study here. In particular, IFD asks whether the instruction improves prediction of the response relative to a reduced context, whereas we ask whether the *given instruction uniquely specifies the task* among other plausible instructions for the same input.

In practice, many instruction datasets contain weakly specified supervision. For a given input X, an output Y may remain plausible under several alternative instructions (see Figure 1). This is especially likely when outputs are short labels, short spans, or generic answers. For example, Alpaca contains tasks with highly constrained target forms such as main-idea extraction and spelling correction, but instruction-tuning corpora also include short-label settings where the same output can remain plausible under several nearby prompts; Dolly similarly mixes strongly constrained extraction/classification prompts with broader assistant-style requests, while Natural Instructions contains explicit task templates whose outputs can still vary in how tightly they are coupled to the wording of the instruction. When this happens, the model can often predict Y from the input alone or from broad task priors, without needing to rely strongly on the instruction. Such examples are therefore easier to fit, but they provide weaker evidence about how instructions should change model behavior. If many of them are included in the training pool, they consume budget without clearly teaching the boundary between neighboring tasks, especially when the

(a) Strongly specified example (high TSS)
Instruction: Find the main idea of the following passage.
Input: Climate change is an increasingly contentious global issue. It is already having a drastic effect on the environment and human lives. We are already seeing an increase in extreme weather events and rising sea levels, both of which have a major impact on the planet.
Output: Climate change is a major global issue that is already seriously affecting the environment and human life.
Why high TSS: The output is tightly tied to the summarization / main-idea instruction, and is much less compatible with nearby alternatives such as *analyze the tone* or *classify the sentiment*.
Dataset grounding: Alpaca includes instruction-following examples such as "Find the main idea of the following passage" and "Analyze the tone of the following sentences," illustrating tasks with specific target forms.

(b) Weakly specified example (low TSS)
Instruction: Classify the sentiment of the sentence.
Input: The movie was fantastic.
Output: Positive.
Why low TSS: The same output remains plausible under several nearby instructions, such as *classify the review tone*, *determine whether the opinion is favorable*, or *label the sentiment polarity*.
Dataset grounding: Dolly contains short-label classification prompts such as "Classify each of the following...", and Natural Instructions includes tightly templated tasks with short valid outputs, making this a realistic low-specificity pattern even when the task descriptions themselves are explicit.

Figure 1: Illustrative high- vs. low-specificity supervision examples grounded in the kinds of prompts visible in Alpaca, Dolly, and Natural Instructions. High-specificity instances tightly couple instruction and output, whereas weakly specified instances admit multiple nearby instructions that can license the same output.

budget is tight or when multiple heterogeneous instruction sources are mixed.

In this paper, we propose the *Task–Specificity Score* (TSS), a simple and model-agnostic measure of how uniquely an output Y is specified by its instruction I among other plausible instructions for the same input X. Rather than asking whether an example is high-quality or easy for the model, TSS asks a different question: *would this output still make sense if the instruction were changed?* Intuitively, an example has high task specificity if Y is strongly compatible with the given instruction and significantly less compatible with reasonable alternative instructions that could have been asked for the same input.

We operationalize this idea by contrasting the log-likelihood of the output under the true instruction with its likelihood under a set of counterfactual instructions generated without access to the output. This yields a pointwise estimate of how informative the instruction–output pairing is, relative to nearby tasks. From an information-theoretic perspective, TSS can be viewed as a Monte Carlo approximation to the conditional mutual information between instruction and output given the input, $I(I; Y \mid X)$, which is otherwise intractable for large language models.

Building on this foundation, we introduce TSS++, a quality-aware contrastive extension designed to be robust on structured instruction corpora. TSS++ combines (i) a normalized, InfoNCE-style contrastive objective computed against *hard* alternative instructions and (ii) an explicit likelihood-based quality term that captures in-distribution fluency. This formulation addresses two practical failure modes of vanilla specificity measures: sensitivity to easy negatives and under-weighting of output quality, which has made simple perplexity filtering surprisingly strong in prior work (Zhou et al., 2023; Raffel et al., 2020).

We evaluate TSS and TSS++ as filtering and weighting mechanisms for instruction tuning under explicit token budgets. Across three widely used instruction datasets, ALPACA (Taori et al., 2023), DOLLY-15K (Conover et al., 2023), and a 20-task subset of NATURAL INSTRUCTIONS (Wang et al., 2022) and three open-weight decoder-only models Gemma (Team et al., 2025), LLaMA (Grattafiori et al., 2024; Meta, 2024), and Qwen (Yang et al., 2025), we measure downstream performance on four benchmarks: ARC-C (Clark et al., 2018), MMLU (Hendrycks et al., 2021), HELLASWAG (Zellers et al., 2019), and TRUTHFULQA (Lin et al., 2022). Our experiments show that selecting or up-weighting high-specificity examples consistently improves budget efficiency and often outperforms strong quality-based baselines, particularly at low retention fractions. Notably, TSS++ remains competitive with perplexity and IFD filters on structured datasets while providing a more principled notion of task informativeness.

Our contributions are threefold:

- We introduce **Task–Specificity Score** (TSS), a principled, instance-level measure of how much an instruction actually constrains the output relative to plausible alternative tasks.
- We propose TSS++, a contrastive, quality-aware extension that combines hard-negative instruction comparisons with likelihood-based quality control.
- Through extensive experiments across models,

datasets, and budget regimes, we demonstrate that task specificity is a complementary and practically useful axis for instruction data selection, improving downstream performance and data efficiency beyond standard quality-based heuristics.

Taken together, our results suggest that not all instruction examples contribute equally to learning how instructions matter. Measuring and exploiting task specificity provides a new lens on supervision quality, one that is especially relevant as instruction tuning continues to scale under finite compute and data budgets.

2 Related Work

Instruction tuning and mixture construction. Instruction tuning has become the dominant paradigm for aligning pretrained LMs to follow natural-language directives, popularized by RLHF-style instruction-following models (Ouyang et al., 2022) and large-scale supervised instruction mixtures (Wei et al., 2022; Wang et al., 2023). Broad instruction corpora such as NATURAL INSTRUCTIONS (Mishra et al., 2022) and subsequent community datasets (e.g., Alpaca (Taori et al., 2023)) motivated training on heterogeneous datasets where the same input can admit many plausible tasks. In practice, these datasets are often mixed using simple heuristics (uniform or size-proportional sampling), which can be effective but do not explicitly measure whether the instruction is actually necessary to recover the labeled output for a given example.

Data filtering/selection for instruction tuning. A long line of work studies dataset selection under limited budgets, including classical cross-entropy difference and domain-adaptation selection (Moore and Lewis, 2010; Axelrod et al., 2011). Recent instruction-tuning settings revisit selection with model-based signals and learned filters: Li et al. (2024b) propose self-guided data selection to trade quantity for quality during instruction tuning, while Li et al. (2024a) study learned "superfilters" that can remove low-utility instruction data at scale. Orthogonally, Xia et al. (2024) frame *targeted* instruction tuning as selecting influential examples for a desired capability via optimizer-aware influence approximations. A complementary survey and taxonomy of instruction-tuning data selection and curation is provided by Liu et al. (2025). Our work fits this literature but focuses on a different axis: isolating *instruction utility* from generic quality, i.e., whether the instruction meaningfully constrains the supervision beyond what other plausible instructions for the same input would support.

Measuring instruction utility beyond quality. Many practical filters for instruction tuning primarily track output "quality" (Zhou et al., 2023; Chen et al., 2024) or in-distributionness (e.g., likelihood/perplexity under a reference model) (Li et al., 2024a), or compare likelihood with vs. without the instruction (instruction-following difficulty style signals) (Li et al., 2024b). These methods are valuable but can conflate (i) high-quality outputs that are compatible with many tasks and (ii) examples where the instruction uniquely determines the label. Our *Task–Specificity Score* instead operationalizes instruction utility through *counterfactual instructions*: we compare the likelihood of the observed output under the true instruction against its likelihood under a set of plausible alternatives for the same input, connecting to contrastive learning views of discrimination among negatives (van den Oord et al., 2019). This explicitly targets when instructions "really matter" for supervision, and is therefore complementary to quality-centric selection (Zhou et al., 2023) and learned filtering methods (Xia et al., 2024; Tu et al., 2025; Xu et al., 2025).

3 Method: Task–Specificity Score (TSS)

Setup. We assume a corpus of instruction–input–output triples $\mathcal{D} = \{(I, X, Y)\}$, where I is a natural-language instruction, X is the input (e.g., question/passage), and Y is the target output (e.g., answer/label/summary). Let p_θ be a pretrained (frozen) language model that assigns conditional probabilities $p_\theta(Y \mid I, X)$ under the standard LM factorization over tokens.

Motivation. Many data selection signals used in instruction tuning are *quality* proxies. For example, perplexity-based filtering prefers examples with high likelihood under the model, and instruction-following difficulty (IFD) (Li et al., 2024b) variants compare likelihood *with* vs. *without* the instruction. These answer questions like: *"Is Y probable under (I, X)?"* or *"Does adding I help predict Y?"* In contrast, we aim to measure *task specificity*: whether the pair (I, Y) is uniquely informative among other plausible instructions that could reasonably be asked for the same input X.

Algorithm 1 Computing Task–Specificity Score (TSS)

Require: Dataset $\mathcal{D} = \{(I_j, X_j, Y_j)\}_{j=1}^{N}$, frozen model p_θ, #alternatives K, retention fraction ρ

for each example $(I, X, Y) \in \mathcal{D}$ **do**
  Sample K alternative instructions $\{I_1, \ldots, I_K\}$ using $\texttt{Prompt}(X)$ (no access to Y)
  $s_{\text{true}} \leftarrow \log p_\theta(Y \mid I, X)$
  for $k = 1$ to K **do**
    $s_k \leftarrow \log p_\theta(Y \mid I_k, X)$
  end for
  $\mathrm{TSS}(I, X, Y) \leftarrow s_{\text{true}} - \log\left(\frac{1}{K}\sum_{k=1}^{K} e^{s_k}\right)$
end for
Threshold $\tau_\rho \leftarrow (1-\rho)$-quantile of $\{\mathrm{TSS}\}$
Retained set $\mathcal{D}_\rho \leftarrow \{(I, X, Y) : \mathrm{TSS}(I, X, Y) \geq \tau_\rho\}$
return $\mathcal{D}_\rho$ (or weights $w = \exp(\lambda \cdot \mathrm{TSS})$)

3.1 Generating plausible alternative instructions

For each triple (I, X, Y), we construct a set of K *plausible alternative instructions* $\{I_1, \ldots, I_K\}$ that could be asked about the same input X *without observing* Y. We use a prompt template[1] that depends only on X, e.g., *"Generate K different instructions a user might give about this input, each asking for a different kind of output."* Denoting this template as $\texttt{Prompt}(X)$, we sample $I_k \sim p_\theta(\cdot \mid \texttt{Prompt}(X))$ for $k = 1, \ldots, K$.

Rationale. These alternatives form a local, model-induced approximation to the distribution over "other tasks one might ask for the same input". Crucially, they are constructed without conditioning on Y.

3.2 Task–Specificity Score (TSS)

Given the true triple (I, X, Y) and its alternatives $\{I_k\}_{k=1}^{K}$, we compare the compatibility of Y with the true instruction against its compatibility under counterfactual instructions. Let $s_{\text{true}} = \log p_\theta(Y \mid I, X)$ and $s_k = \log p_\theta(Y \mid I_k, X)$. We define the Task–Specificity (see Algorithm 1):

$$\mathrm{TSS}(I, X, Y) = s_{\text{true}} - \log\left(\frac{1}{K}\sum_{k=1}^{K} e^{s_k}\right) = \log \frac{p_\theta(Y \mid I, X)}{\frac{1}{K}\sum_{k=1}^{K} p_\theta(Y \mid I_k, X)}. \quad (1)$$

Interpretation. A high TSS indicates that Y is substantially more likely under the true instruction than under other plausible instructions for the same input. A low (or negative) TSS suggests that Y remains similarly plausible under many alternative tasks, implying weakly specified supervision.

[1] Refer A for exact prompt template.

3.3 Information-Theoretic view

Conceptually, we care about how much information the instruction provides about the output beyond the input alone, captured by conditional mutual information $I(I; Y \mid X)$. Pointwise, this resembles $\log \frac{p(Y|I,X)}{p(Y|X)}$, but $p(Y \mid X)$ is intractable for LMs. TSS can be viewed as a Monte Carlo approximation where $p_\theta(Y \mid X)$ is estimated by averaging over alternative instructions: $p_\theta(Y \mid X) \approx \frac{1}{K}\sum_{k=1}^{K} p_\theta(Y \mid I_k, X)$, which yields Eq. 1.

3.4 Using TSS for Budgeted Filtering and Weighting

Once $\mathrm{TSS}(I, X, Y)$ is computed for each training instance, we use it in two ways.

Filtering under a token budget. Given a retention fraction $\rho \in (0, 1]$, we keep the top-ρ fraction of examples by TSS. Equivalently, let τ_ρ be the $(1-\rho)$-quantile of the TSS scores; retain $\mathcal{D}_\rho = \{(I, X, Y) \in \mathcal{D} : \mathrm{TSS}(I, X, Y) \geq \tau_\rho\}$. This implements budgeted selection (e.g., $\rho = 0.05$ for 5% of data/tokens).

Importance weighting. Alternatively, we define a nonnegative weight per example as $w = \exp(\lambda \cdot \mathrm{TSS})$ with scale $\lambda \geq 0$, and train with a weighted loss $\mathbb{E}[w \cdot \ell_\theta(I, X, Y)]$. Exponentiation provides a smooth, monotone mapping from score to positive weights and emphasizes high-TSS examples; in practice, λ is tuned and weights may be clipped for stability.

3.5 TSS++: Quality-aware contrastive specificity

On structured instruction corpora (e.g., NI/Dolly-style), two practical issues can reduce the effectiveness of vanilla TSS: (i) *easy* or stylistically mismatched alternatives can make the denominator too small, inflating TSS, and (ii) specificity alone can under-emphasize output *quality* (fluency / in-distributionness), where strong PPL filters often excel. We address this with TSS++, which combines (a) a contrastive (InfoNCE-style) (van den Oord et al., 2019) specificity term computed against *hard* alternatives, and (b) an explicit quality term.

Hard alternatives. Instead of using only random samples, we construct a candidate dataset $\mathcal{C}(X)$ of alternative instructions and keep the most confusable ones. In practice, $\mathcal{C}(X)$ can be generated alternatives, retrieved in-distribution instructions using KNN over embeddings of original instruction, or a

Algorithm 2 TSS++: Quality-Aware Contrastive Task Specificity

Require: Example (I, X, Y), frozen model p_θ, candidate dataset $\mathcal{C}(X)$, hard set size m, temperature τ, quality weight α

Compute normalized scores $s(I', X, Y) = \frac{1}{|Y|} \log p_\theta(Y \mid I', X)$ for all $I' \in \mathcal{C}(X)$

Select hard alternatives $\mathcal{H}(X) \leftarrow$ TOP-m instructions in $\mathcal{C}(X) \setminus \{I\}$ by $s(I', X, Y)$

$s^+ \leftarrow s(I, X, Y)$

$\mathrm{TSS}_{\mathrm{NCE}} \leftarrow \log \dfrac{\exp(s^+/\tau)}{\exp(s^+/\tau) + \sum_{I^- \in \mathcal{H}(X)} \exp(s(I^-, X, Y)/\tau)}$

$\mathrm{TSS{+}{+}}(I, X, Y) \leftarrow \mathrm{TSS}_{\mathrm{NCE}} + \alpha \cdot s^+$

return $\mathrm{TSS{+}{+}}(I, X, Y)$ (for filtering or weighting)

of both; we then select the hard set $\mathcal{H}(X) \subset \mathcal{C}(X)$ as those with highest average log-likelihood of Y under the alternative instruction.

Normalized log-likelihood. We use length-normalized log-likelihood $s(I, X, Y) = \frac{1}{|Y|} \log p_\theta(Y \mid I, X)$ to reduce sensitivity to output length, especially when Y is short.

Contrastive specificity (InfoNCE form). Given temperature $\tau > 0$ and hard alternatives $\mathcal{H}(X)$, define

$$\begin{aligned}\mathrm{TSS}_{\mathrm{NCE}}(I, X, Y) &= \log \frac{\exp\big(s(I, X, Y)/\tau\big)}{\exp\big(s(I, X, Y)/\tau\big) + \sum_{I^- \in \mathcal{H}(X)} \exp\big(s(I^-, X, Y)/\tau\big)} \\ &= -\log\Bigg(1 + \sum_{I^- \in \mathcal{H}(X)} \exp\Big(\big(s(I^-, X, Y) - s(I, X, Y)\big)/\tau\Big)\Bigg).\end{aligned} \tag{2}$$

This normalization makes the score depend on *relative* compatibility between the true instruction and hard alternatives. Hard negatives are essential: if alternatives are too easy (very low likelihood), the contrast provides little discrimination.

Quality-aware TSS++. Finally, we add an explicit quality term (the in-distribution likelihood under the true instruction) (see Algorithm 2):

$$\mathrm{TSS{+}{+}}(I, X, Y) = \mathrm{TSS}_{\mathrm{NCE}}(I, X, Y) + \alpha \cdot s(I, X, Y), \tag{3}$$

where $\alpha \geq 0$ trades off instruction-specificity (contrastive) against output quality (likelihood). We use TSS++ in the same way as TSS: filter top-ρ examples or compute weights $w = \exp(\lambda \cdot \mathrm{TSS{+}{+}})$.

4 Experimental Setup

4.1 Data

We study three instruction-following corpora: Alpaca[2] (Taori et al., 2023), Dolly-15K[3] (Conover et al., 2023), and NI-20 (Wang et al., 2022; Kadasi et al., 2025) (a subset of Natural Instructions with 20 task types), refer Table 5, 6 and 7 for more details. Each training instance is a triple (I, X, Y) consisting of an instruction, an input, and a target response.

4.2 Models

We evaluate across three open-weight pretrained LLM families: Gemma-3-1B-PT[4] (Team et al., 2025), LLaMA-3.2-1B[5] (Grattafiori et al., 2024; Meta, 2024), and Qwen3-0.6B[6] (Yang et al., 2025). We use the corresponding 12B instruct variants of these model families to generate alternative instructions, while using the 1B base models for TSS/TSS++ scoring. We fine-tune each 1B model under controlled budget settings.

4.3 Budget Settings

We evaluate budgeted instruction tuning using retention fractions[7] $\rho \in \{0.05, 0.15, 0.45, 0.75, 1.0\}$, where ρ denotes the fraction of the original training set retained for a run. If $\mathcal{D}$ is the full training set and $|\mathcal{D}|$ is the number of training examples in that set, then a budget ρ corresponds to keeping approximately $\rho|\mathcal{D}|$ examples.

For each dataset–model pair and each ρ, we first construct a retained set of size $\rho|\mathcal{D}|$ using the corresponding selection criterion (Random, PPL, IFD, TSS, or TSS++). We then train in one of two ways: (i) **subset training**, where all retained examples are used uniformly; or (ii) **weighted subset training**, where the same retained examples are assigned per-example weights derived from their scores (e.g., exponential weighting for TSS or TSS++), so higher-scoring examples contribute more to the training loss.

4.4 Data Selection and Weighting Variants

For each dataset–model pair and each budget ρ, we construct one retained training set using the corresponding selection criterion. For **Random**, we sample a single uniform random subset at the target retention fraction ρ using a fixed seed; we do not average over multiple random draws in the main

[2] https://huggingface.co/datasets/yahma/alpaca-cleaned

[3] https://huggingface.co/datasets/databricks/databricks-dolly-15k

[4] https://huggingface.co/google/gemma-3-1b-pt

[5] https://huggingface.co/meta-llama/Llama-3.2-1B

[6] https://huggingface.co/Qwen/Qwen3-0.6B-Base

[7] We use the term *retention fraction* to emphasize that each method deterministically selects and trains on a fixed subset of the original supervision, rather than resampling data dynamically during training.

results. We compare TSS and TSS++ against the following data construction strategies (definitions are provided in Section 3):

- **Random**: uniform random subset at each budget ρ.
- **PPL**: perplexity-based ranking for subset selection (Li et al., 2024a).
- **IFD**: instruction-following difficulty ranking (Li et al., 2024b).

For implementation details, refer to Appendix D.

4.5 Evaluation and Experimental Setting

We study budgeted instruction tuning across retention fractions $\rho \in \{0.05, 0.15, 0.45, 0.75, 1.0\}$ on three training datasets (ALPACA, DOLLY-15K, and NI-20) and three decoder-only models (Gemma, Llama, and Qwen). All models are trained with the same fine-tuning recipe within a given dataset–model setting: one epoch of LoRA-based supervised fine-tuning, learning rate 5×10^{-5}, batch size 4, gradient accumulation 4, maximum sequence length 1024, weight decay 0.01, and warmup ratio 0.03. We use AdamW, bfloat16 training, gradient clipping with maximum norm 1.0, and a cosine learning-rate schedule after linear warmup. The weighted variants use the same retained subset as their corresponding filtering method, but scale the per-example loss by an exponential function of the score; in the rest of the paper, we denote these by TSS (E) and TSS++ (E).

All trained checkpoints are evaluated using the LM-Eval Harness[8] (Biderman et al., 2024) on four downstream benchmarks: ARC-C (Clark et al., 2018), MMLU (Hendrycks et al., 2021), HELLASWAG (Zellers et al., 2019), and TRUTHFULQA (Lin et al., 2022). We report the standard benchmark score for each dataset and also their unweighted sum (**SUM**), which serves as our primary aggregate metric. Although these benchmarks are not identical in format, they are all reported on comparable percentage-style scales in our evaluation setup, making SUM a simple summary of overall downstream performance; higher is better.

5 Results

We now evaluate how data selection affects downstream performance under explicit retention budgets. All four evaluation benchmarks (ARC-C, MMLU, HELLASWAG, and TRUTHFULQA) are reported on comparable percentage-like scales, so we summarize overall performance with an unweighted aggregate. For readability, our tables report **SUM**, the sum of the four benchmark scores; equivalently, this is a constant rescaling of the corresponding unweighted average (**AVG** $=$ **SUM**$/4$). Thus, method rankings are identical under SUM and AVG, and we retain SUM throughout for consistency with the result logs and plots.

5.1 Low-budget operating points: 5% and 15% retention

We begin with the most practically relevant regime, where only a small fraction of the available supervision can be retained. Tables 1 and 2 summarize, for each (model, dataset) pair, the best-performing strategy at $\rho = 0.05$ and $\rho = 0.15$, together with the improvement over the Random subset baseline at the same budget.

5% retention At $\rho = 0.05$, the best strategy improves over Random in **8/9** settings, with a mean gain of **+0.703** SUM points (**+0.176** AVG). The largest gain is **+1.646** SUM for Gemma on NI-20 (**132.137** vs. 130.491; Table 1). Across the nine model–dataset combinations, TSS++ is the most frequent winner (**6/9**), followed by TSS (**2/9**), while Random wins only **1/9**. This concentration of wins is the clearest evidence that instruction-specific scoring is most useful in the tightest budget regime.

15% retention. At $\rho = 0.15$, the best strategy improves in **6/9** settings, with a mean gain of **+1.253** SUM points (**+0.313** AVG). The largest gain is **+2.881** SUM for Qwen on NI-20 (**154.765** vs. 151.884), achieved by TSS with exponential weighting (Table 2). The winner set is more diverse than at 5%: Random wins **3/9**, TSS wins **3/9**, TSS++ wins **2/9**, and a quality baseline wins **1/9** (PPL on Llama–DOLLY). Taken together, Tables 1 and 2 show that instruction-specific selection is strongest when the budget is most constrained, while quality-oriented heuristics become more competitive as more data is retained.

5.2 Full-budget trends and optimal retention fractions

The representative budget sweeps in Figure 2 (Gemma–NI-20, Llama–ALPACA, and Qwen–DOLLY) at Appendix C help explain the low-budget results and the crossovers at larger ρ.

[8] https://github.com/EleutherAI/lm-evaluation-harness

Model	Dataset	Random	Best	Δ	Best method
Gemma	Alpaca	132.758	**132.902**	+0.144	TSS
Gemma	Dolly	130.091	**130.950**	+0.859	TSS++ (E)
Gemma	NI-20	130.491	**132.137**	+1.646	TSS
Llama	Alpaca	141.921	**142.704**	+0.783	TSS++ (E)
Llama	Dolly	140.270	**140.573**	+0.303	TSS++
Llama	NI-20	140.642	**140.642**	+0.000	Random
Qwen	Alpaca	156.415	**157.162**	+0.747	TSS++ (E)
Qwen	Dolly	152.514	**153.582**	+1.068	TSS++ (E)
Qwen	NI-20	154.376	**155.153**	+0.777	TSS++ (E)

Table 1: Budgeted subset selection at 5% retention. "Random" uses a random subset of equal size; "Best" is the highest SUM among all selection/weighting strategies at this budget. Δ is computed w.r.t. Random. SUM is the unweighted sum of ARC-C, MMLU, HellaSwag, and TruthfulQA.

Model	Dataset	Random	Best	Δ	Best method
Gemma	Alpaca	134.250	**134.250**	+0.000	Random
Gemma	Dolly	130.724	**132.231**	+1.507	TSS (E)
Gemma	NI-20	129.941	**131.834**	+1.893	TSS
Llama	Alpaca	141.825	**144.603**	+2.778	TSS++
Llama	Dolly	140.234	**140.646**	+0.412	PPL
Llama	NI-20	140.717	**140.717**	+0.000	Random
Qwen	Alpaca	158.580	**159.215**	+0.635	IFD
Qwen	Dolly	152.303	**154.110**	+1.807	TSS++ (E)
Qwen	NI-20	151.884	**154.765**	+2.881	TSS (E)

Table 2: Budgeted subset selection at 15% retention (same conventions as Table 1).

A first observation is that the curves are *not necessarily monotonic* in ρ. This is expected in our setup for two reasons. First, each retained subset is constructed independently at each budget, so the $\rho = 0.45$ set is not a strict nested superset of the $\rho = 0.15$ set. Second, increasing the fraction can admit lower-utility or more weakly specified examples, which may dilute the training signal even though more data is used. Thus, our sweeps should be interpreted as comparisons between *different operating points* on the budget–utility trade-off, rather than as a cumulative data-scaling curve.

The second observation is that the best operating point often occurs at an *intermediate* retention fraction rather than at $\rho = 1.0$. Table 3 reports the best configuration across all budgets and strategies; in **8/9** cases, the best result occurs at $\rho < 1.0$. For example, Llama on ALPACA peaks at **147.304** SUM with TSS++ (E) at $\rho = 0.45$, outperforming full-dataset tuning by **+7.916** SUM. Similarly, Qwen on ALPACA peaks at **159.215** SUM with IFD at $\rho = 0.45$, a gain of **+2.372** SUM over full-dataset tuning. Even Gemma on NI-20 reaches its best overall score already at $\rho = 0.05$, confirming that for some structured pools, a very small but carefully chosen subset can outperform larger retained sets.

To make this pattern explicit, Table 3 reports the best retention fraction as well for each model–dataset pair. The table shows that the optimal ρ is highly dataset- and model-dependent, but rarely equal to 1.0: only one combination (Llama–DOLLY) peaks at the full retained set.

A third observation is methodological: the representative sweeps in Figure 2 show that, in all three displayed cases, an instruction-specific method (TSS or TSS++) forms the top-performing curve and attains the best operating point among all compared baselines. This is especially clear for Gemma–NI-20 and Llama–ALPACA, where the gains of TSS/TSS++ persist across a substantial portion of the budget range rather than at a single isolated point.

5.3 Winner map across budgets

Figure 3 at Appendix C provides a compact heatmap view of the winning *method family* at each budget. Rows correspond to (model, dataset) pairs, columns correspond to ρ, and each cell indicates whether the winner belongs to Random, a quality baseline (PPL/IFD), TSS, or TSS++.

Two trends emerge. First, the heatmap makes

Model	Dataset	Zero-shot	Full SFT	Best overall	Δ vs Full	Method (budget)
Gemma	Alpaca	129.923	135.176	**136.089**	+0.913	Random (0.75)
Gemma	Dolly	129.923	131.509	**132.526**	+1.017	Random (0.75)
Gemma	NI-20	129.923	128.977	**132.137**	+3.160	TSS (0.05)
Llama	Alpaca	127.493	139.388	**147.304**	+7.916	TSS++ (E) (0.45)
Llama	Dolly	127.493	138.191	**141.979**	+3.788	PPL (1.0)
Llama	NI-20	127.493	132.912	**140.717**	+7.805	Random (0.15)
Qwen	Alpaca	154.218	156.843	**159.215**	+2.372	IFD (0.45)
Qwen	Dolly	154.218	152.394	**154.110**	+1.716	TSS++ (E) (0.15)
Qwen	NI-20	154.218	151.595	**155.153**	+3.558	TSS++ (E) (0.05)

Table 3: Best configuration across all budgets and selection strategies (SUM). "Full SFT" corresponds to training on the entire dataset; "Zero-shot" is evaluated without instruction tuning. "Method (budget)" reports the winning selection heuristic and the retained fraction.

Benchmark	Random	Best	Δ
ARC-C	32.594	**33.959**	+1.365
MMLU	34.831	**37.423**	+2.592
HellaSwag	48.357	**48.994**	+0.637
TruthfulQA	25.459	**26.928**	+1.469
SUM	141.241	**147.304**	+6.063

Table 4: Per-benchmark breakdown for a representative high-gain setting: Llama on Alpaca at 45% retention. "Best" corresponds to TSS++ (E). All four benchmarks improve simultaneously, indicating that gains are not driven by a single dataset.

the *budget dependence* of the winners explicit: instruction-specific methods occupy a large fraction of the low-to-mid budget region, while Random and quality-based methods appear more frequently only at higher retention fractions. Second, the crossovers are structured rather than noisy. The same row can transition from TSS++ at low ρ to TSS or a quality baseline at larger ρ, which is consistent with the view that specificity is most valuable when the budget is tight, whereas fluency, coverage, and general quality become more important as more supervision is admitted.

5.4 Where do the gains come from? Benchmark-level breakdown

To check whether the gains in aggregate score are driven by a single benchmark or reflect broader generalization, we analyze a representative high-gain setting: Llama on ALPACA at $\rho = 0.45$. Table 4 reports the per-benchmark scores for the winning configuration.

The winning TSS++ configuration improves all four benchmarks simultaneously (ARC-C, MMLU, HellaSwag, and TruthfulQA), showing that the increase in aggregate score is not due to overfitting one benchmark at the expense of the others. This makes the main message of the section stronger: under explicit token budgets, instruction-specific selection does not merely reshuffle performance across tasks, but can improve generalization more broadly.

6 Conclusion

We introduced **Task–Specificity Score** (TSS), a simple, model-based measure of whether a target output Y is *uniquely compatible* with the given instruction I among plausible alternative instructions for the same input X. TSS operationalizes instruction dependence as a contrast between the likelihood under the true instruction and an empirical estimate of $p(Y \mid X)$ formed by averaging over alternatives. We further proposed TSS++, which strengthens specificity estimation by focusing on hard alternatives via an InfoNCE-style contrastive objective and by adding an explicit quality term to remain competitive with strong quality-oriented filters. Across three instruction pools and three open LLMs, we find that prioritizing high-specificity examples improves downstream performance under tight retention budgets and complements quality-based selection methods, highlighting that *instruction sensitivity* is a distinct and practically useful signal beyond generic likelihood. More broadly, our results suggest that data curation for instruction tuning should consider not only whether outputs are "easy" or "likely," but also whether the instruction meaningfully constrains the supervision signal.

7 Limitations

Our study has several limitations. First, TSS and TSS++ are *model-dependent*: scores are computed using a frozen LM, so the ranking can change with the choice of scoring model, its calibration,

and its prompt/template. While we evaluate across three decoder-only models and observe consistent trends, we do not claim model-invariant specificity estimates. Second, TSS relies on generating or retrieving plausible alternative instructions. If the alternative set is low-quality (e.g., repetitive, implausible, or overly generic), TSS may overestimate specificity; conversely, if alternatives are too strong or off-distribution, it may underestimate specificity. Although TSS++ mitigates "easy-negative" effects via hard negatives, the method remains sensitive to the alternative construction pipeline (generation prompts, retrieval corpus, embedding model, and hyperparameters). Third, scoring is computationally non-trivial: computing per-example likelihoods for K alternatives (and hard-negative mining) can be expensive for large corpora, even though it requires only forward passes. This cost may limit applicability in very large-scale instruction pools without additional approximation (e.g., caching, smaller scoring models, or reduced K). Fourth, our experiments focus on three instruction pools (Alpaca, Dolly-15K, NI-20, three small open models, and four downstream benchmarks. The conclusions may not fully transfer to larger models, other domains (e.g., multimodal or tool-use data), or alternative evaluation suites. In particular, **SUM** is an unweighted aggregate over a small benchmark set; different benchmark mixes or weighting schemes may change the observed trade-offs. Finally, TSS targets instruction-specificity and does not directly guarantee factuality, harmlessness, or alignment of the resulting tuned models. Data selection may improve generalization under a budget but does not replace safety-specific training or evaluation.

8 Ethical Considerations

This work proposes a data scoring and selection framework for instruction tuning. Because our approach can *amplify* certain subsets of data, it may also amplify biases present in the underlying instruction pools or the scoring model. For example, if the frozen LM assigns systematically higher likelihood to particular dialects, writing styles, or demographic topics, TSS-based selection could disproportionately retain such examples. We recommend applying standard dataset governance practices (documentation, bias analysis, and auditing) and monitoring distributional shifts introduced by selection. In addition, our method leverages existing public instruction datasets that may contain problematic content (toxicity, stereotypes, sensitive information, or licensing restrictions). We do not introduce new personal data collection, but we encourage users of TSS/TSS++ to respect dataset licenses and to apply content filtering where appropriate. From a misuse perspective, improved data efficiency can lower the barrier to tuning models for harmful purposes. While TSS is intended for research on data quality and instruction dependence, we advise that deployments incorporate safety mitigation (e.g., refusal training, red teaming, and usage policies) and that release artifacts avoid enabling targeted harmful fine-tuning. Finally, there are environmental and resource considerations: although TSS uses only forward passes, large-scale scoring can still incur substantial compute and energy consumption. We report compute details and encourage practitioners to reuse cached scores, prefer smaller scoring models when possible, and reduce K or hard-negative mining cost under limited resources.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *Preprint*, arXiv:2405.14782.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*.

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. *Preprint*, arXiv:1803.05457.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly:

Introducing the world's first truly open instruction-tuned llm.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Pritam Kadasi, Abhishek Upperwal, and Mayank SIngh. 2025. Adapt: Learning task mixtures for budget-constrained instruction tuning. *Preprint*, arXiv:2512.04555.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024a. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14255–14273, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. 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, Mexico City, Mexico. Association for Computational Linguistics.

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

Ziche Liu, Rui Ke, Yajiao Liu, Feng Jiang, and Haizhou Li. 2025. Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6595–6611, Albuquerque, New Mexico. Association for Computational Linguistics.

Meta. 2024. Llama 3.2: Revolutionizing edge ai and vision with open models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. Released September 25, 2024.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

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

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pages 19920–19930. Curran Associates, Inc.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Zeao Tu, Xiangdi Meng, Yu He, Zihan Yao, Tianyu Qi, Jun Liu, and Ming Li. 2025. ResoFilter: Fine-grained synthetic data filtering for large language models through data-parameter resonance analysis. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5429–5443, Albuquerque, New Mexico. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language

models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, and 16 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: Selecting influential data for targeted instruction tuning. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. 2025. Stronger models are not always stronger teachers for instruction tuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4392–4405, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Kumar Ravikumar. 2022. First is better than last for language data influence. In *Advances in Neural Information Processing Systems*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Qiu Jiantao, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. 2025. Harnessing diversity for important data selection in pretraining large language models. In *The Thirteenth International Conference on Learning Representations*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

A Prompts

Alternative Instructions Generator Prompt

SYSTEM (when input text is available).

```
Generate exactly {k} distinct task
    instructions for the input below.
Each instruction must:
- Be answerable using only that input (no
    external documents).
- Ask for a different type of output than
    the others.
- Be a single sentence, at most 25 words.

Return a numbered list (1., 2., ...) with
    one instruction per line.
Do not include any explanations, headers,
    or extra text.
```

USER

```
Input:
{input_text}
```

SYSTEM (when input text is empty).

```
Given the original task instruction below,
    generate exactly {k} distinct
    alternative
task instructions that do NOT require any
    extra input text.

Each alternative instruction must:
- Be about the same general topic or domain
    as the original instruction.
- Ask for a different type of output or
    perspective (not just a paraphrase).
- Be a single sentence, at most 25 words.

Return a numbered list (1., 2., ...) with
    one instruction per line.
Do not include any explanations, headers,
    or extra text.
```

USER

```
Original instruction:
{original_instruction}
```

B Datasets

Category	#Tasks	#Instances	#Tokens
Translation	394	1,182,213	72,549,385
Question Answering	207	470,108	106,180,992
Program Execution	90	433,157	35,066,354
Sentiment Analysis	75	253,432	32,670,340
Question Generation	83	230,103	56,131,362
Text Matching	43	173,171	14,178,609
Text Categorization	46	154,556	11,876,178
Commonsense Classification	24	130,524	2,237,835
Toxic Language Detection	40	115,584	5,148,102
Fill in The Blank	22	93,210	13,687,063
Textual Entailment	27	92,651	4,886,841
Information Extraction	34	91,850	6,710,148
Text Completion	21	86,145	8,631,760
Sentence Perturbation	15	80,789	2,472,566
Title Generation	19	80,696	20,869,481
Wrong Candidate Generation	27	73,546	9,763,136
Sentence Composition	20	72,496	5,066,324
Question Understanding	16	63,448	3,239,390
Pos Tagging	10	62,118	2,583,225
Summarization	16	59,200	43,445,932

Table 5: Natural Instructions dataset statistics by task category.

Category	#Tasks	#Instances	#Tokens
general_qa	–	2191	278,298
summarization	–	1188	413,617
closed_qa	–	1773	443,893
open_qa	–	3742	247,093
creative_writing	–	709	155,342
classification	–	2136	115,900
information_extraction	–	1506	445,951
brainstorming	–	1766	142,254

Table 6: Category-wise statistics for Dolly-15k. Dolly does not provide explicit task IDs in the same way as Natural Instructions, so the #Tasks column is left undefined.

Category	#Tasks	#Instances	#Tokens
–	–	51,760	6,453,067

Table 7: Corpus-level statistics for Alpaca-cleaned. Alpaca does not provide explicit category or task identifiers in the same way as task-structured instruction datasets, so the corresponding fields are left undefined.

C Main Content Figures

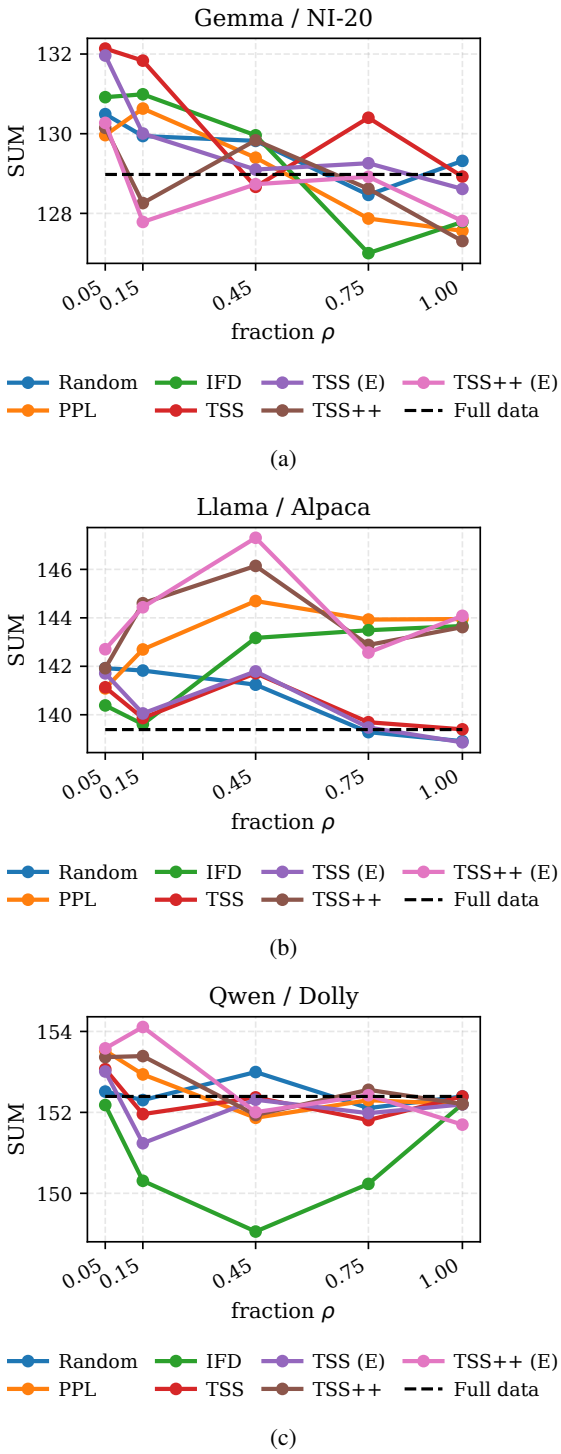

Figure 2: **Budget sweeps.** X-axis: retention fraction $\rho \in \{0.05, 0.15, 0.45, 0.75, 1.0\}$. Y-axis: SUM. Plot one curve per method (Random, PPL, IFD, TSS, TSS (E), TSS++, TSS++ (E)).

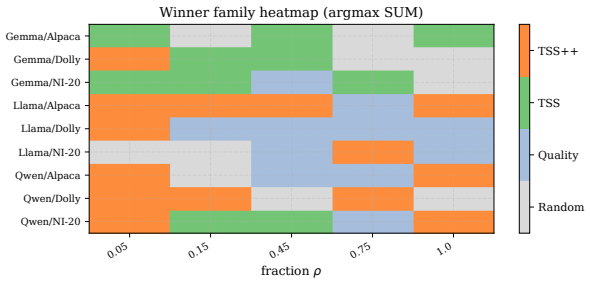

Figure 3: **Winner map across budgets.** A heatmap with rows as (model, dataset) and columns as ρ; each cell indicates the winning method family (Random vs Quality vs TSS vs TSS++).

D Implementation Details

All selection and weighting scores are computed using a *separate frozen scoring model*, i.e., a model used only for scoring examples and not updated during subset construction. Concretely, for each model family we use the corresponding 1B base model as the scoring model for TSS/TSS++, while alternative instructions are generated using the corresponding 12B instruct model from the same family. To keep scoring tractable, we set the number of generated alternatives to $K = 4$ per instance when constructing TSS/TSS++ scores. This choice reflects a compute–quality trade-off: larger K gives a better approximation to the alternative-instruction distribution, but increases scoring cost linearly; in our setting, $K = 4$ provided a practical balance between diversity of alternatives and total runtime.