SCAR: Efficient Instruction-Tuning for Large Language Models via Style Consistency-Aware Response Ranking

Anonymous ACL submission

Abstract

Recent studies emphasize that manually ensuring a consistent response style and maintaining high data quality in training sets can significantly improve the performance of fine-tuned Large Language Models (LLMs) while reducing the number of training examples needed. However, the precise definition of style and the relationship between style, data quality, and LLM performance remains unclear. This research identifies two key stylistic elements in responses: linguistic form and instructional surprisal. We find that, among training data of comparable quality, higher consistency in these response elements leads to better LLM performance. Inspired by this, we introduce Style Consistency-Aware Response Ranking (SCAR), which automatically prioritizes instruction-response pairs in the training set based on their response stylistic consistency. By selecting the most style-consistent examples, sometimes as few as 0.7% of the full dataset, the fine-tuned LLMs can match or even surpass the performance of models trained on the entire dataset in coding and open-ended question-answering benchmarks. Code and data are available at `https://anonymous.4open.science/r/SCAR-0233/`.

1 Introduction

Instruction-following Large Language Models (LLMs), such as GPT-3.5 and GPT-4 (Achiam et al., 2023), have demonstrated strong generalization across diverse language tasks (Chung et al., 2022; Ouyang et al., 2022). These models are trained in stages: unsupervised pre-training on large text corpora, followed by supervised fine-tuning (SFT) on instruction-response pairs and additional optimization stages (Bai et al., 2022).

Recent studies, such as AlpaGasus (Chen et al., 2024) and LIMA (Zhou et al., 2024), demonstrate that carefully curated, smaller datasets can outperform larger ones in improving LLM SFT performance. AlpaGasus finds that smaller datasets with higher quality scores, rated by GPT-4 for helpfulness or correctness, outperform significantly larger ones when used to fine-tune high-capacity LLMs. The *Superficial Alignment Hypothesis*, proposed in LIMA, suggests that pre-trained language models already possess the necessary knowledge, and fine-tuning is to guide the model toward specific response styles, thus not requiring large amounts of data. LIMA achieves notable performance with only 1,000 high-quality instruction-response pairs, optimized for *consistent style* by human experts. However, this hypothesis raises three open questions: (i) *What key elements constitute response styles that impact LLM SFT?* (ii) *How does data quality (i.e., helpfulness, correctness) relate to style consistency in influencing efficient SFT?* (iii) *Can we develop an automatic method that measures stylistic elements to curate smaller, stylistically consistent datasets for more efficient SFT at a lower cost, without relying on human experts?*

Text style is shaped by **consistent choices** across various linguistic elements (Kang and Hovy, 2021; Karlgren, 2004), such as lexical, syntactic, and semantic features (DiMarco and Hirst, 1993). Our empirical studies have identified two key stylistic factors within responses that significantly affect LLM SFT performance: **Linguistic Form** and **Instructional Surprisal**. **Linguistic Form** comprises the lexical and syntactic choices that define how a response is presented, independent of its meaning. Empirically, this includes transitional and functional word usage, sentence structure, punctuation patterns, layout features (e.g., headers, bullet points), etc. **Instructional Surprisal**, in our definition, measures how surprising a response is, focusing on the semantic relationship between response and instruction. We demonstrate that *among training datasets with similarly helpful and accurate responses, those responses with greater consistency in linguistic form and instructional surprisal produce better-performing LLMs.*

Achieving style consistency is challenging, even for human experts. We found that datasets containing LLM-generated responses with consistent styles can significantly outperform human-crowdsourced data in enhancing LLM performance. Therefore, we introduce **S**tyle **C**onsistency-**A**ware Response **R**anking (SCAR), a novel ranking-based model that prioritizes instruction-response pairs with high stylistic consistency and superior data quality. SCAR is trained on LLM-synthesized and human-crowdsourced datasets to reward responses with higher style consistency regarding linguistic form and instructional surprisal. Enhanced with representation learning, SCAR can better distinguish between these two elements and prioritize aspects that improve LLM performance. Experiments show that by selecting the most style-consistent examples, sometimes as little as 0.7% of the original dataset, fine-tuned LLMs can match or surpass the performance of models trained on full datasets like OCTOCODER-15.5B (Muennighoff et al., 2023) and OLMO-7B-SFT (Groeneveld et al., 2024) on coding (HumanEval; Chen et al. 2021) and open-ended question answering (AlpacaEval; Dubois et al. 2023) benchmarks.

In summary, our contributions are two-fold:

(I) We identify linguistic form and instructional surprisal as critical response style elements, and demonstrate that within training datasets with comparable helpfulness and accuracy, responses exhibiting higher consistency in linguistic form and instructional surprisal yield better LLMs.

(II) We develop SCAR, a ranking method that selects high-quality, stylistically consistent examples from style-inconsistent datasets. When selecting training data for efficient SFT, SCAR **outperforms leading data selection baselines**, enabling LLMs trained on small subsets (0.7–25% of original data) to **match or exceed full-dataset performance**.

2 Impact of Styles on LLM Fine-tuning

In this section, we study two research questions: i) What key elements in response style can influence LLM SFT? and ii) How do style consistency and data quality impact LLM performance?

RQ1: What Factors Constitute Response Style

Through empirical analysis of stylistic differences between synthetically generated and human-written instruction-tuning data, we identified two key sets of stylistic features in responses that significantly influence LLM alignment performance.

Linguistic Form refers to the structure of language, including how words and sentences are organized and interact (Fabb, 2001; Chomsky, 1957; Jurafsky, 2000). In our context, it denotes elements that shape the presentation of a response, mostly independent of semantics, such as transitional and functional word usage, tone, sentence structure, punctuation patterns, and layout features (e.g., headers, bullet points), etc. For example, GPT-3.5-TURBO responses often follow a consistent structure, using bullet points and similar transitional phrases across responses, whereas human responses, authored by diverse individuals, tend to exhibit greater variation in the linguistic elements.

Instructional Surprisal measures how surprising a response is in addressing a given instruction, focusing on the semantic alignment of its content (solutions, ideas, and approaches) with the instruction. For example, when asked about sorting algorithms, GPT-3.5-TURBO consistently provides predictable solutions like merge sort or quick sort, while human responses show a range of surprisal–from conventional approaches to unexpected choices like StoogeSort or novel answers.

RQ2: Influence of Style Consistency and Data Quality on LLM Performance

We collect both human-written and synthetic data in coding and general open-ended domains, and conduct stylometric and quality analyses on this data. Following this, we fine-tune base LLMs using this data to explore how style consistency and data quality influence LLM SFT performance.

We control style variations to create three dataset types–human-written, referenced, and direct–to explore how linguistic form and response surprisal impact LLM performance. In the coding domain, we collect 10,000 human-written instruction-response pairs from StackExchange[1], an online platform that includes 11 million pairs of coding questions and answers. We use the LIMA dataset, including 1,000 human-generated examples, for the general domain. Additionally, we generate two synthetic response types with controlled styles: "referenced" and "direct." "Referenced" responses are crafted by a chat-LLM that rewrites human responses to retain their semantic meaning, similar to the method in Yang et al. (2024). This process retains the surprisal levels of human responses but alters their linguistic

[1]https://stackexchange.com/

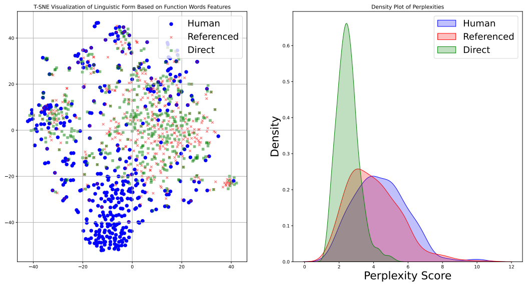

Figure 1: (Left) T-SNE plot showing embeddings of the linguistic forms of human and GPT-3.5-TURBO responses to LIMA instructions. (Right) Density plot of perplexity detailing the surprisal levels of the responses.

form. We also filter out examples where "referenced" responses deviate significantly from human responses in surprisal metrics, reducing the datasets to 944 StackExchange and 407 LIMA examples. In contrast, the chat-LLM generates "direct" responses to the remaining instructions without any references, potentially producing different semantics, thereby significantly varying their surprisal levels compared to human-referenced responses. In Appendix F, we provide examples to illustrate the similarities and differences among these three style variants.

We also isolate the effects of data quality on LLM performance by using three chat-LLMs with different capabilities to generate synthetic "referenced" and "direct" datasets. The models employed are GPT-3.5-TURBO, LLAMA2-70B-CHAT, and LLAMA2-13B-CHAT (Touvron et al., 2023), with GPT-3.5-TURBO being the most advanced, followed by LLAMA2-70B-CHAT and LLAMA2-13B-CHAT, according to the arena-leaderboard (Zheng et al., 2024). We find hallucinations that occur during the LLM generation of "referenced" and "direct" responses can significantly affect quality of the resulting synthetic data.

Stylometric Analysis. *To analyze the linguistic form of human and synthetic responses*, we employ six authorship attribution metrics (Tripto et al., 2023; Zheng and Jin, 2023) that capture non-semantic features. These include the Type Token Ratio (TTR) (Templin, 1957), Measure of Textual Lexical Diversity (MTLD) (McCarthy, 2005) for functional words, Flesch score (Kincaid et al., 1975), average sentence length, and the frequency of punctuation and layout features (e.g., bullet points and headers). Higher TTR and MTLD values indicate greater lexical diversity, while a higher Flesch score suggests improved readability. We identify functional words in the response using a lexicon based on heuristic POS-tagging rules. *To assess instructional surprisal*, we compute perplexity, a well-established metric for measuring text surprisal (Oh and Schuler, 2023; Goodkind and Bicknell, 2018), denoted as PPL$(y|x)$, using META-LLAMA-3-8B (AI@Meta, 2024).

T-SNE (Van der Maaten and Hinton, 2008) plots (Figure 1, left) show that embeddings of GPT-3.5-TURBO-generated "referenced" and "direct" responses cluster tightly in the center, indicating that both synthetic response types share consistent and similar *linguistic forms*. These embeddings are created by vectorizing six authorship metrics and the unigrams of functional words. Conversely, human responses are more dispersed in the outer region, showing lower consistency. Figure 1 (right) shows "direct" responses have a more skewed perplexity distribution towards lower values, indicating higher consistency in *instructional surprisal* compared to both "referenced" and human ones.

Standard deviations (Std.) of TTR and perplexity for different response types are listed in Table 1, with additional linguistic form and text surprisal metrics detailed in Table 5 (Appendix B.2). We observe human responses have higher Std. values regarding TTR, perplexity and other metrics compared to synthetic responses, and "referenced" responses show a higher perplexity Std. than "direct" responses. The Std. values of these metrics across "referenced" and "direct" responses from LLAMA2-70B-CHAT, LLAMA2-13B-CHAT, and GPT-3.5-TURBO indicate synthetic responses from **all** these LLMs have higher consistency in both stylistic elements than human ones.

Data Quality Analysis. We evaluate a sample of 100 examples from each dataset using GPT-4-1106-PREVIEW. We rate the scores for two data quality metrics, *helpfulness* and *correctness*, using the adjusted prompt from the automatic data evaluator ICE-Score (Zhuo, 2024) for the coding domain and AlpaGasus (Chen et al., 2024) for the open-ended domain, and then calculate the average scores across the samples. Higher scores indicate better quality. Table 1 reveals that in the coding domain, GPT-3.5-TURBO-generated responses match the quality of human-written ones, while other LLMs produce lower-quality data. In the open domain, LLAMA2-70B-CHAT and GPT-3.5-TURBO responses are comparable in quality to human-written responses, whereas LLAMA2-13B-CHAT responses are of slightly lower quality.

Impact on LLM Performance. We evaluate the CODELLAMA-7B model fine-tuned with

Data Curation Methods	StackExchange			LIMA		
	Stylometric Analysis	Data Quality	CodeLlama-7B Performance	Stylometric Analysis	Data Quality	Meta-Llama-3-8B Performance
	Std. TTR ↓ / Std. PPL($y|x$)↓	Helpfulness / Correctness	Avg. Pass@1 / Avg. Pass@10	Std. TTR↓ / Std. PPL($y|x$)↓	Helpfulness / Correctness	L.C. WinRate
Human Response	24.23 / 0.33	3.29 / 3.70	26.56 / 41.63	20.49 / 1.53	3.86 / 4.14	1.93
GPT-3.5-turbo						
Referenced	8.16 / 0.33	3.44 / 3.70	29.82 / 46.89	18.43 / 1.52	3.79 / 4.00	3.64
Direct	8.14 / 0.30	3.32 / 3.45	31.00 / 47.12	16.06 / 0.64	3.91 / 4.16	5.67
Llama2-70b-chat						
Referenced	11.90 / 0.36	3.14 / 3.54	29.82 / 44.03	16.51 / 1.45	3.89 / 4.11	3.96
Direct	13.52 / 0.28	3.18 / 2.71	30.89 / 45.31	15.63 / 0.42	3.85 / 4.22	6.25
Llama2-13b-chat						
Referenced	7.46 / 0.27	2.65 / 2.68	26.61 / 41.91	13.64 / 1.19	3.75 / 3.89	3.77
Direct	8.86 / 0.28	1.85 / 1.70	26.42 / 40.00	14.22 / 0.38	3.29 / 3.48	6.22

Table 1: Performance comparison of CodeLlama-7b and Meta-Llama-3-8B fine-tuned on training sets curated using different methods and various LLMs, along with data quality and stylometric analysis for the training sets.

LoRA (Hu et al., 2021) on various datasets using HumanEval (Python) (Chen et al., 2021) and MultiPL-E (Java, JavaScript, C++) (Cassano et al., 2023) benchmarks. For the coding domain, we report average Pass@1 and average Pass@10 execution accuracies across all coding questions spanning four programming languages. We measure the length control win rate (L.C. WinRate) (Dubois et al., 2024) by comparing responses from the LoRA fine-tuned Meta-Llama-3-8B with those from GPT-4-preview-1106 on 2500 open-domain instructions from AlpacaEval[2]. We use Llama-3-70b-chat (AI@Meta, 2024) as our automatic evaluator for its cost-effectiveness ($0.9 per evaluation). This evaluator is comparable with GPT-4 evaluators in correlating with human judgment, surpassing human-to-human agreement (67.5 vs. 65.7), given the agreement tests on AlpacaEval.

When comparing synthetic responses of similar or slightly different quality generated from capable chat-LLMs, "direct" responses outperform their "referenced" counterparts in downstream LLM SFT tasks through higher instructional surprisal consistency. Both synthetic types exhibit greater consistency in both stylistic elements, thereby outperforming human-authored data. However, style consistency alone cannot compensate for substantial quality deficits. This is evidenced by a notable exception in coding tasks, where Llama2-13b-chat's "direct" responses, despite having higher style consistency, achieve poorer fine-tuning outcomes due to their significantly lower quality scores (1.8) compared to both "referenced" responses (2.6) and human data (3.5).

Takeaway. The analysis reveals several insights:

(I) ***Linguistic form*** and ***instructional surprisal*** inherent in the response styles of the training data significantly influence the LLM SFT performance.

(II) LLM-generated responses show higher style consistency than human ones, with "direct" responses showing the greatest consistency in ***linguistic form and instructional surprisal***.

(III) Enhancing data quality and ensuring response style consistency within a dataset both contribute to improved LLM SFT performance. Among datasets with shared instructions and similar quality, those with responses exhibiting higher style consistency yield better LLM performance.

3 Style Consistency-Aware Ranking

Inspired by these findings, we develop a Style Consistency-Aware Ranker to select training examples with consistent response styles, improving LLM SFT performance.

Ranking Objective. Given a dataset $\mathcal{D} = \{(x_i, y_i^d, y_i^r, y_i^h)\}_{i=1}^N$, where x_i represents the instruction, y_i^d and y_i^r are the "direct" and human "referenced" responses from chat-LLMs, respectively, and y_i^h represents the human response. ***We aim to learn a ranking function*** $R(x, y)$ ***that assigns higher scores to high-quality responses adhering to the consistent style of a specific LLM's outputs.*** The objective for each instance is to learn the ranking function:

$$\mathcal{L}_r(x, y^d, y^r, y^h) = \sum_{(y^a, y^b) \in \mathcal{P}} \max(0, \alpha - R_{\boldsymbol{\theta}}(x, y^a) + R_{\boldsymbol{\theta}}(x, y^b)) \quad (1)$$

$$\text{s.t.} \quad \min(f(x, y^a),\ f(x, y^b)) > \sigma \quad (2)$$

where $\mathcal{P} = \{(y^d, y^r), (y^r, y^h), (y^d, y^h)\}$ represents the set of desired pairwise orderings, based on the findings from Section 2, that "direct" responses are more consistent in surprisal levels than "referenced" ones, "referenced" responses are more consistent in linguistic form than human data, and "direct" responses are more consistent than human data in both stylistic feature types. The margin α ensures the difference in the ranking scores assigned by $R_{\boldsymbol{\theta}}(x, y)$, while the quality measure func-

[2]https://github.com/tatsu-lab/alpaca_eval/

tion $f(x, y)$ evaluates the quality (e.g., helpfulness, correctness) of the response y given the instruction x. The quality measure function f can be implemented using strong LLMs such as GPT-3.5 or GPT-4 with a prompt, as in Chen et al. (2024), to evaluate the helpfulness and correctness of the answers and average these scores to obtain the final quality score. The quality threshold σ ensures the ranker only rewards responses that are **both style-consistent and high-quality**.

Reward Function. The reward function $R_{\boldsymbol{\theta}}(x, y)$ is modelled as a neural network that takes representations of instructional surprisal $\mathbf{v}_c \in \mathbb{R}^{1\times M}$ and linguistic form $\mathbf{v}_p \in \mathbb{R}^{1\times M}$, and computes a scalar reward score using a multi-layer perceptron (MLP):

$$\begin{aligned} R_{\boldsymbol{\theta}}(x, y) &= \text{MLP}_r([\mathbf{v}_p; \mathbf{v}_c]) \\ \mathbf{v}_p &= \text{Max-Pool}(\mathbf{V}_y) \\ \mathbf{v}_c &= \text{MLP}_c([\mathbf{V}_x^0; \mathbf{V}_y^0]) \end{aligned} \tag{3}$$

Our experiments show that linguistic form has a minimal influence on instructional surprisal compared to semantic content and is significantly less dependent on the instruction. These findings motivate us to adopt disentangled modeling strategies. For linguistic form, we capture surface-level features through max pooling over the response sequence $\mathbf{V}_y$, independent of the instruction. For instructional surprisal, drawing on prior work that models surprisal as a text–context (Michaelov et al., 2023; Karampiperis et al., 2014) relation, we capture multi-dimensional semantic alignment by passing the concatenated [CLS] embeddings of the instruction and response through an MLP. We use an encoder, such as RoBERTa-base (Liu et al., 2019), to generate the sequence representations $\mathbf{V}$. Refer to the Appendix B.4 and B.5 for independence tests and background on surprisal modelling.

Style Representation Learning. Accurately capturing distinct representations for linguistic form ($\mathbf{v}_p$) and instructional surprisal ($\mathbf{v}_c$) is challenging, as these features can still become entangled during the learning process, even with our specialized separation design. To address this, we leverage observed similarities: the linguistic form of "referenced" responses is more similar to "direct" responses than to human responses, and the instructional surprisal of "referenced" responses is closer to that of human responses than to "direct" ones, as shown in Figure 1. We introduce a regularization term using triplet margin losses to enforce these similarity patterns:

$$\begin{aligned} &\mathcal{L}_{rl}(x, y^d, y^r, y^h) = \\ &\lambda_p \max\{0, d(\mathbf{v}_p^d,\mathbf{v}_p^r) - d(\mathbf{v}_p^r, \mathbf{v}_p^h) + \beta_p\} \\ &+\lambda_c \max\{0, d(\mathbf{v}_c^h,\mathbf{v}_c^r) - d(\mathbf{v}_c^d, \mathbf{v}_c^h) + \beta_c\} \end{aligned} \tag{4}$$

where $d(\mathbf{v}_i, \mathbf{v}_j) = \|\mathbf{v}_i - \mathbf{v}_j\|_2$ is the distance function and β values are the margins.

Final Loss Function. The final loss function combines the ranking loss and the representation learning losses: $\mathcal{L}_{scar} = \mathcal{L}_r + \mathcal{L}_{rl}$

Ranking and Filtering. After training reward function $R_{\boldsymbol{\theta}}(x, y)$, it ranks instruction-response pairs (x, y) in a held-out dataset. The top $k\%$ of examples with the highest scores are selected to create a high-quality style-consistent subset for fine-tuning LLMs. This filtered dataset is expected to ***improve the performance of fine-tuned LLMs on target tasks more than using the entire original dataset***.

4 Experiments

We train SCAR using data from the ***coding*** and ***open-ended question-answering*** domains to select examples for LLM SFT from the full dataset in these same domains.

Ranker Data. We collect instructions for SCAR training and evaluation, which include 10,000 randomly selected examples from StackExchange for the code domain, and 6,000 instructions from a combination of 5,000 random Dolly (Conover et al., 2023) data samples and the full LIMA dataset. Dolly is a human-curated dataset with 15,000 high-quality instruction-response pairs. We create the data by pairing instructions with human responses and the "referenced" and "direct" responses generated by GPT-3.5-turbo, as described in Section 2. Due to budget limitations, we use GPT-3.5-turbo to rate the helpfulness and correctness of responses according to the constraint in Eq.(2).

LLM SFT Data. SCAR and other baselines select data from two sources, held out from the ranking training data. These sources provide diverse but style-inconsistent examples: ***i) Human-Crowdsourced Data***, curated by many authors, making it diversified and naturally style-inconsistent. ***ii) Mixed Synthetic Data***, generated by GPT-3.5-turbo using various system prompts, reflecting the practical use of multiple open-source synthetic datasets to enhance diversity.

For the code domain, human-written data comes from a sample of 20,000 crowdsourced StackExchange examples.

The mixed synthetic data comprises 20,000 examples, sourced evenly from: i) 5000 StackExchange instructions with "direct" responses, ii) 5000 StackExchange instructions with "referenced" responses, iii) 5,000 coding examples curated using Evol-Instruct (Luo et al., 2023) by Zan et al. (2023), and iv) 5,000 coding examples generated using Self-Instruct (Wang et al., 2023b). The instructions cover Python, Java, JavaScript, and C++. For Self-Instruct, we use GPT-3.5-TURBO to generate responses in the target programming languages.

For the open-ended domain, human-written data comes from 10,000 Dolly examples, held out from the Dolly examples used for ranker training.

Mixed synthetic data includes 10,000 examples, evenly sourced from: i) 2,500 held-out Dolly instructions with "direct" answers, ii) 2,500 Dolly instructions with "referenced" answers, iii) 2500 open-domain examples using Self-Instruct by LaMini (Wu et al., 2023b), and iv) examples curated using Evol-Instruct from Xu et al. (2023).

Data Selection and LLM SFT. The data selection methods sample 50%, 25%, and 12.5% of coding-domain data to fine-tune CODELLAMA-7B, and 50%, 25%, and 10% of open-domain data to fine-tune META-LLAMA-3-8B. Both LLM trainings use LoRA due to computational constraints.

LLM Evaluation. We use HumanEval and Multip-E for coding evaluation, reporting the Avg. Pass@(1+10) $= \frac{(Avg.Pass@1+Avg.Pass@10)}{2}$ across four languages for fine-tuned CODELLAMA-7B. For general tasks, we use AlpacaEval and report the L.C. WinRate of outputs from fine-tuned META-LLAMA-3-8B compared to GPT-4-PREVIEW-1106, as in Section 2.

Data Selection Baselines. We compare SCAR in two settings with 7 baselines: i) RANDOM: Randomly select examples. ii)PERPLEXITY (Albalak et al., 2024): Select examples with the lowest response perplexity (PPL($y|x$)) computed using META-LLAMA-3-8B. III) SUPERFILTERING (Li et al., 2024): Select the most challenging examples for LLMs with the highest Instruction-Following Difficulty (IFD) score. Here, we compute IFD as $\frac{\text{PPL}(y|x)}{\text{PPL}(y)}$ using META-LLAMA-3-8B. IV) HUMAN FEEDBACK RANKING (HFR): Use the same ranker architecture as SCAR trained on 10,000 stack-exchange-paired (Lambert et al., 2023) examples annotated given human preference (each instruction paired with positive and negative responses) for coding domain and 6000 human preference examples from Anthropic RLHF data (Bai et al., 2022) for the general domain. V) ALPAGASUS (Chen et al., 2024): Select data based on response quality scores rated by GPT-3.5-TURBO, consistent with the rating method used in our ranker. VI) DIVERSITY: Apply k-means clustering to diversify examples by selecting randomly from each cluster, a method commonly used in active learning (Li and Haffari, 2023; Li et al., 2023c; Zhdanov, 2019). VII) LONGEST: Select examples with longest response token lengths (Zhao et al.). VIII) SCAR (ID): SCAR trained on in-domain (ID) data (e.g., code) and selects examples within the same domain. IX) SCAR (OOD): SCAR trained on in-domain data and select examples from an out-of-domain (OOD) dataset. For instance, SCAR(OOD) is trained on the code domain and selects data from the open domain or vice versa.

4.1 Main Results and Discussion

Effectiveness of SCAR-Selected Data. As in Figure 2, SCAR(ID) can enhance SFT performance while lowering computational costs. LLMs fine-tuned on only 25% and 10% of SCAR(ID)-selected data achieve comparable or superior performance to models trained on full datasets in coding and general domains, respectively.

SCAR(ID) and SCAR(OOD) consistently outperform other methods in data selection for fine-tuning LLMs, with SCAR(OOD) slightly lagging behind SCAR(ID) due to cross-domain generalization. Some baselines show unstable performance. SUPERFILTERING performs poorly in the coding domain. We observe it may assign high IFD scores to erroneous examples in crowdsourced coding data of varying quality. PERPLEXITY and ALPAGASUS-selected data result in similar LLM performance trends. However, their performance is inferior to SCAR(ID), which we attribute to their lack of style consistency. Traditional active learning methods like RANDOM and DIVERSITY sampling prove less effective, as our style-inconsistent target scenario inherently incorporates diversity, limiting their additional benefits. HFR's underperformance across most scenarios suggests that training the ranker on inconsistent human preferences from diverse authors may impair its ability to select optimal training data. Notably, LONGEST performs comparably to our method in open-domain synthetic data selection, though inferior elsewhere. This aligns with our style consistency framework, as length serves as a strong style indicator, with Evol-Instruct re-

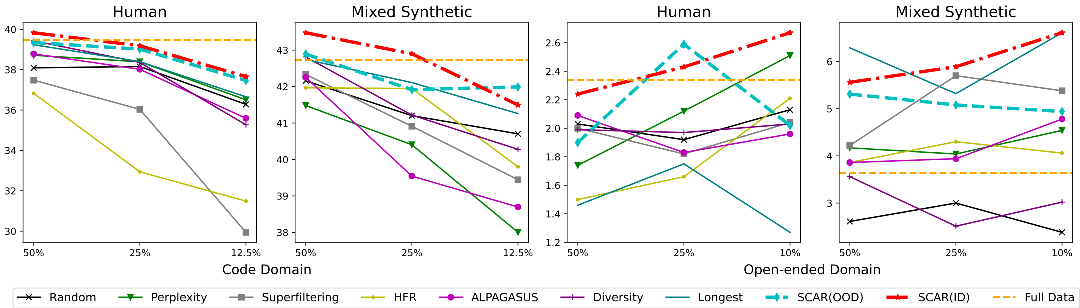

Figure 2: The performance of LLMs fine-tuned on human and synthetic data subsets of various sizes in code and open domains, sampled with different data selection approaches.

	Std. TTR	Std. PPL	Helpful	Correct
StackExchange				
100%	21.48	1.80	2.84	2.68
50%	16.78	1.61	3.02	3.01
25%	14.85	1.61	2.78	2.72
12.5%	14.29	1.94	2.67	2.77
Dolly				
100%	30.96	65.70	3.95	3.91
50%	28.43	54.32	3.98	3.99
25%	24.74	49.51	3.96	3.93
10%	23.73	39.58	3.98	3.99

Table 2: Stylometric and quality analysis of data subsets selected by SCAR(ID) from the full human-crowdsourced StackExchange and Dolly datasets.

sponses consistently being longer.

Impact of Data Sizes. Figure 2 shows that in the coding domain, using fewer data selected by various methods usually lowers LLM performance. However, in the open-ended domain, most methods can select fewer synthetic data to fine-tune LLMs that outperform those trained on the full dataset. With SCAR(ID), reducing data consistently improves LLM performance in the open domain. This suggests that while dataset size, diversity, and style consistency can all benefit LLM SFT, their optimal balance varies across different scenarios.

Stylometric and Data Quality Analysis of SCAR-Selected Data. Table 2 shows that SCAR(ID) improves style consistency in the selected Dolly data, reflected by consistently lower TTR and perplexity standard deviation compared to the full dataset. However, for code data, while the TTR standard deviation decreases, the perplexity standard deviation increases when selecting smaller subsets (25%, 12.5%), suggesting that differentiating semantic surprisal features in code is challenging. This may explain the sudden performance drop in LLMs fine-tuned on these smaller code subsets. Moreover, our method preserves average data quality (helpfulness, correctness), as rated using GPT-4-1106-PREVIEW, comparable to the full dataset, likely due to the use of the data quality constraint in Eq. (2) during ranker training.

Effectiveness of SCAR on Open-Source LLMs. We fine-tune OLMO-7B (Groeneveld et al., 2024) and STARCODER-15.5B (Li et al., 2023b) on subsets of their publicly available SFT datasets.

OLMO-7B	Data Sizes	320k	10k	5k	2.5k
	L.C. WinRate	3.86	5.37	5.64	4.08
STARCODER-15.5B	Data Sizes	13k	10k	5k	2.5k
	Avg. Pass@(1+10)	37.85	39.69	40.09	40.14

Table 3: L.C. WinRate for OLMO-7B and Avg. Pass@(1+10) for STARCODER-15.5B fine-tuned on original (320k, 13k) and subset sizes (10k, 5k, 2.5k).

Specifically, we select 2.5k, 5k, and 10k examples from the *allenai/tulu-v2-sft-mixture* (320k) and *bigcode/guanaco-commits* (13k) datasets. These subsets consist of a mixture of synthetic and human-generated data, selected using the SCAR(ID) method. We then compare their performance to the official checkpoints, OLMO-7B-SFT and OCTOCODER-15.5B (Muennighoff et al., 2023), which were instruction-tuned on the full datasets. Table 3 shows that SCAR-selected subsets significantly boost performance, achieving these results with only 0.7% to 20% of the original data, as measured by L.C. WinRate on AlpacaEval and average Pass@(1+10) on HumanEval and MultiPL-E. Further evaluation of OLMO-7B variants on diverse benchmarks (Table 14, Appendix C.7)–including ARC-Challenge (Clark et al., 2018), TruthfulQA (Lin et al., 2022), HellaSwag (Zellers et al., 2019) and MMLU (Hendrycks et al.)–demonstrates that **all** our subset-fine-tuned OLMO-7B outperform the full 320k-trained model in average performance across various LLM capabilities.

4.2 Ablation Study

To evaluate the effectiveness of SCAR(ID) components, we compare the full ranker training setting (Full, GPT-3.5) against variations without the quality constraint in Eq. (2) (w/o con, GPT-3.5), without representation learning in Eq. (4) (w/o rl, GPT-3.5), and without "referenced" responses during training (w/o ref, GPT-3.5). We also generate synthetic data to train the ranker using LLAMA2-13B-CHAT (Full, Llama2-13b), LLAMA2-70B-CHAT (Full, Llama2-70b), LLAMA-3-70B-CHAT (Full, Llama3-70b), and LLAMA2-13B-CHAT without using quality constraint (w/o con, Llama2-13b).

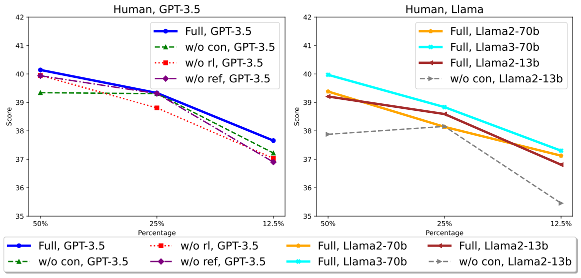

Figure 3: Performance of LLMs fine-tuned on subsets of human-written data selected by SCAR(ID), trained with different configurations and synthetic data sources (e.g., GPT-3.5, Llama).

Style Representation Learning. Figure 3 shows that removing the representation learning loss (w/o rl, GPT-3.5) or excluding "referenced" responses (w/o ref, GPT-3.5) only slightly reduces LLM performance in the code domain. The objective in Eq. (4) is likely satisfied even without the loss because "referenced" responses provide an intermediate style during training, which is why we set a low coefficient (0.1) for this loss. However, excluding "referenced" responses significantly degrades performance in the open domain (Table 17, Appendix D.1) and disrupts the optimization of Eq. (4). Table 18, Appendix D.2 further analyses the representation learning results.

Data Quality Constraint. Figure 3 (2nd) shows that removing the data quality constraint in Eq. (2) significantly worsens the performance of LLMs fine-tuned on human-crowdsourced data when SCAR is trained on lower-quality datasets, such as LLAMA2-13B-CHAT-generated responses (w/o con, Llama2-13b), compared to using the constraint (Full, Llama2-13b). In this case, SCAR tends to select style-consistent but erroneous or unhelpful examples from LLM SFT data with varying quality(e.g. crowdsourced data). However, in other cases, removing the quality constraint has minimal impact on data selection performance.

LLMs for Generating SCAR Training Data. Figure 3 shows that using LLAMA-generated synthetic data for training SCAR slightly reduces fine-tuned LLM performance compared to GPT-3.5-TURBO-generated data, but the impact is more severe with LLAMA2-13B-CHAT-generated data. This is likely because the quality constraint filters out 90% of low-quality LLAMA2-13B-CHAT examples, limiting the ranker's generalization ability. Style misalignment between the LLAMA and GPT-3.5-TURBO data may also affect data selection performance when selecting mixed synthetic GPT-3.5-TURBO data.

5 Related Work

Instruction-Tuning Data Selection. Instruction-tuning trains LLMs to follow complex instructions in various contexts (Wei et al., 2021; Sanh et al., 2021). Data are sourced from human-curated examples (Wang et al., 2022b; Zhou et al., 2024) and LLM outputs (Xu et al., 2023; Wang et al., 2022a). Studies (Zhou et al., 2024; Chen et al., 2024; Li et al., 2024, 2023a; Lu et al., 2023; Liu et al.) show that smaller, high-quality datasets can outperform significantly larger ones in boosting LLM performance. LIMA uses expert human curation for stylistic consistency (Zhou et al., 2024), while AlpaGasus (Chen et al., 2024) utilizes LLMs to assess data quality. Other methods select effective examples based on Instruction Following Difficulty scores (Li et al., 2024, 2023a), diversity metrics (Lu et al., 2023; Bukharin and Zhao, 2023), or response length (Zhao et al.).

Automatic Authorship Detection. Our method relates to authorship detection studies. Traditional authorship detection used lexical features like TTR, MTLD, and Flesch readability scores (Tripto et al., 2023; Zheng and Jin, 2023). Recent focus has shifted to distinguishing human and machine-generated texts using advanced neural networks to analyze styles at the corpus (Mitchell et al., 2023; Su et al., 2023) or the sentence levels (Zeng et al., 2024, 2023; Wang et al., 2023a; Zeng et al.). The studies (Xu and Sheng, 2024; Su et al., 2023; Wang et al., 2023a; Mitchell et al., 2023; Wu et al., 2023a), like ours, show perplexity effectively differentiates between human and machine styles.

6 Conclusion

Our empirical study demonstrates that, among training datasets with comparable helpfulness and correctness, those with higher consistency in linguistic form and instructional surprisal significantly enhance the performance of fine-tuned LLMs. Building on this insight, we propose SCAR, a ranking method designed to measure and select stylistically consistent training data for LLM fine-tuning. Our experiments show that LLMs fine-tuned on small subsets of the original dataset–using as little as 0.7% of the data selected by SCAR–can outperform models trained on the full datasets. Moreover, SCAR consistently outperforms other data selection baselines in LLM fine-tuning.

Limitations

Reducing the training dataset size can potentially introduce biases. To address this concern, we discuss two types of bias: fairness bias and lexical diversity bias.

Fairness Bias. SCAR-selected subsets may exhibit some degree of toxicity and sentiment polarity towards certain demographic and occupational groups. However, the overall fairness performance is generally **comparable to, or even better than,** that achieved with full-data training and other selection methods. While fairness biases may persist, this issue is not unique to SCAR but is a challenge faced by all LLMs. Refining selection criteria to further minimize such biases remains a promising direction for future work. For a detailed analysis, see Tables 19 and 20 at Appendix E.1.

Lexical Diversity Bias. Lexical diversity in instructions and responses is evaluated separately using TTR and MTLD. Compared to the full dataset and other subsets, SCAR-selected instructions exhibit a slightly lower TTR, with a more noticeable reduction in responses. However, MTLD scores, which reflect deeper, length-independent lexical richness, remain comparable to those of the full dataset and subsets selected by other selection baselines. This indicates that while SCAR reduces some surface-level lexical variation in responses (as reflected by TTR), it does not significantly compromise the overall richness of vocabulary in either instructions or responses. Importantly, since instruction-level diversity plays a more critical role in LLM fine-tuning performance (Lu et al., 2023; Bukharin and Zhao, 2023), SCAR-selected subsets retain the diversity that matters most. The minor reduction in TTR does not pose a significant issue, as demonstrated by SCAR's strong performance in our extensive experiments. See Table 21 at Appendix E.2 for detailed analysis.

References

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

AI@Meta. 2024. Llama 3 model card.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. 2024. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Alexander Bukharin and Tuo Zhao. 2023. Data diversity matters for robust instruction tuning. *arXiv preprint arXiv:2311.14736*.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, et al. 2023. Multipl-e: a scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Noam Chomsky. 1957. *Syntactic Structures*. De Gruyter Mouton, Berlin, Boston.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

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*.

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.

Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. Bold: Dataset and metrics for measuring biases in open-ended language generation. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 862–872.

Chrysanne DiMarco and Graeme Hirst. 1993. A computational theory of goal-directed style in syntax. *Computational Linguistics*, 19(3):451–500.

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

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint*, arXiv:2305.14387.

N. Fabb. 2001. Language and literature. In Neil J. Smelser and Paul B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*, pages 8292–8297. Pergamon, Oxford.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th workshop on cognitive modeling and computational linguistics (CMCL 2018)*, pages 10–18.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

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

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Daniel Jurafsky. 2000. Speech and language processing.

Dongyeop Kang and Eduard Hovy. 2021. Style is NOT a single variable: Case studies for cross-stylistic language understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2376–2387, Online. Association for Computational Linguistics.

Pythagoras Karampiperis, Antonis Koukourikos, and Evangelia Koliopoulou. 2014. Towards machines for measuring creativity: The use of computational tools in storytelling activities. In *2014 IEEE 14th International Conference on Advanced Learning Technologies*, pages 508–512. IEEE.

Jussi Karlgren. 2004. The wheres and whyfores for studying textual genre computationally. In *AAAI Technical Report (7)*, pages 68–70. Citeseer.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

Nathan Lambert, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush. 2023. Huggingface h4 stack exchange preference dataset.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

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

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023a. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023b. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Zhuang Li and Gholamreza Haffari. 2023. Active learning for multilingual semantic parser. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 633–639.

Zhuang Li, Lizhen Qu, Philip R Cohen, Raj Tumuluri, and Gholamreza Haffari. 2023c. The best of both worlds: Combining human and machine translations for multilingual semantic parsing with active learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9511–9528.

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.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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. In *The Twelfth International Conference on Learning Representations*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Philip M McCarthy. 2005. *An assessment of the range and usefulness of lexical diversity measures and the potential of the measure of textual, lexical diversity (MTLD)*. Ph.D. thesis, The University of Memphis.

JA Michaelov, MD Bardolph, CK Van Petten, BK Bergen, and S Coulson. 2023. Strong prediction: Language model surprisal explains multiple n400 effects. neurobiology of language, 1–71. Technical report, Retrieved 2023-04-25, from https://doi.org/10.1162/nol a 00105 doi: 10.1162 ....

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Niklas Muennighoff, Qian Liu, Armel Randy Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. 2023. Octopack: Instruction tuning code large language models. In *The Twelfth International Conference on Learning Representations*.

Byung-Doh Oh and William Schuler. 2023. Why Does Surprisal From Larger Transformer-Based Language Models Provide a Poorer Fit to Human Reading Times? *Transactions of the Association for Computational Linguistics*, 11:336–350.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

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

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.

Mildred C Templin. 1957. *Certain language skills in children: Their development and interrelationships*, volume 10. JSTOR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Nafis Tripto, Adaku Uchendu, Thai Le, Mattia Setzu, Fosca Giannotti, and Dongwon Lee. 2023. HANSEN: Human and AI spoken text benchmark for authorship analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13706–13724, Singapore. Association for Computational Linguistics.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Eva Vanmassenhove, Dimitar Shterionov, and Matthew Gwilliam. 2021. Machine translationese: Effects of algorithmic bias on linguistic complexity in machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2203–2213, Online. Association for Computational Linguistics.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *ACL*.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023a. Seqxgpt: Sentence-level ai-generated text detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1144–1156.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *The 61st*

Annual Meeting Of The Association For Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi DQ Bui, Junnan Li, and Steven CH Hoi. 2023c. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922*.

Jason Wei, Maarten Bosma, Vincent 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. In *International Conference on Learning Representations*.

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023a. Llmdet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2113–2133.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2023b. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *arXiv preprint arXiv:2304.14402*.

A.D. Wyner. 1978. A definition of conditional mutual information for arbitrary ensembles. *Information and Control*, 38(1):51–59.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv e-prints*, pages arXiv–2304.

Zhenyu Xu and Victor S Sheng. 2024. Detecting ai-generated code assignments using perplexity of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23155–23162.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1028–1043, Bangkok, Thailand. Association for Computational Linguistics.

Daoguang Zan, Ailun Yu, Bo Shen, Jiaxin Zhang, Taihong Chen, Bing Geng, Bei Chen, Jichuan Ji, Yafen Yao, Yongji Wang, and Qianxiang Wang. 2023. Can programming languages boost each other via instruction tuning? *Preprint*, arXiv:2308.16824.

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.

Zijie Zeng, Shiqi Liu, Lele Sha, Zhuang Li, Kaixun Yang, Sannyuya Liu, Dragan Gaševic, and Guanliang Chen. Detecting ai-generated sentences in human-ai collaborative hybrid texts: Challenges, strategies, and insights.

Zijie Zeng, Lele Sha, Yuheng Li, Kaixun Yang, Dragan Gašević, and Guangliang Chen. 2024. Towards automatic boundary detection for human-ai collaborative hybrid essay in education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22502–22510.

Zijie Zeng, Lele Sha, Yuheng Li, Kaixun Yang, Dragan Gašević, and Guanliang Chen. 2023. Towards automatic boundary detection for human-ai hybrid essay in education. *arXiv preprint arXiv:2307.12267*.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. In *Forty-first International Conference on Machine Learning*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20.

Fedor Zhdanov. 2019. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Wanwan Zheng and Mingzhe Jin. 2023. A review on authorship attribution in text mining. *Wiley Interdisciplinary Reviews: Computational Statistics*, 15(2):e1584.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Terry Yue Zhuo. 2024. Ice-score: Instructing large language models to evaluate code. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2232–2242.

Appendix Table of Contents

A Implementation Details

A.1 Model Training Configurations

We fine-tune the META-LLAMA-3-8B and CODELLAMA-7B models using LoRA, a parameter-efficient tuning method, on NVIDIA A100 GPUs to minimize computational costs. Both models undergo three training epochs with a learning rate of 2×10^{-5}, using a cosine learning rate scheduler and a warm-up ratio of 0.03. Training is performed with BF16 and TF32 precision modes enabled. For META-LLAMA-3-8B, we employ a single GPU with a batch size of 2, while for CODELLAMA-7B, two GPUs are used with the same batch size, incorporating LoRA parameters set to $r = 8$ and $\alpha = 16$. For the OpenAI models, we adopt `gpt-3.5-turbo-0125` and `gpt-4-1106-preview` as our default configurations. We set the maximum input length for the LLMs to 2048 tokens.

The SCAR ranker is trained with a learning rate of 2×10^{-5} for up to 20 epochs, using early stopping based on validation performance. For code domain tasks, we utilize CODET5P-110M-EMBEDDING (Wang et al., 2023c) for contextual representation encoding, while for open-domain tasks, we employ ROBERTA-BASE (Liu et al., 2019). When curating StackExchange examples for the ranker and LLM training, we ensure quality by selecting instructions containing code blocks and answers with ratings above 2.

A.2 Prompt for Generating Referenced Response

The prompt used to rewrite the human response to generate the "referenced" response is as follows:

```
### Reference Answer:
{human response}

### Background
You are a knowledgeable AI assistant.
Above is the reference answer. Below is an instruction that describes
 a task. Given the reference answer, write a response that
appropriately completes the request.
Please keep the semantics of the reference answer unchanged in your
response, while pretending as if you have never seen the reference
answer, when crafting your final response.

### Instruction:
{instruction}

### Response:
```

A.3 Prompt for Generating Direct Response

The prompt instruction to generate "direct" response is as follows:

```
### Background
You are a knowledgeable AI assistant.
Below is an instruction that describes a task. Please write a
response that appropriately completes the request.

### Instruction:
{instruction}

### Response:
```

B Extended Analysis of Style Effects on LLM Fine-Tuning Performance

B.1 Extended Analysis of LLM Performance on Coding Tasks

Table 4 presents the detailed results for the coding tasks mentioned in Table 1, providing a comprehensive breakdown of the Pass@1 and Pass@10 metrics for each task, rather than just the average scores.

Table 4 reveals that "direct" responses outperform "referenced" responses across most programming benchmarks, suggesting that generating answers without mirroring human semantic content yields better results for coding tasks. For instance, GPT-3.5-TURBO-generated "direct" achieves a Pass@1 of 33.00% on the HumanEval benchmark, compared to 28.58% for GPT-3.5-TURBO-generated "referenced," and similar trends are observed across Java, JavaScript, and C++ benchmarks. Human responses also lag behind "direct" and "referenced" responses, indicating that synthetic data can offer better stylistic consistency, which can boost LLM SFT performance. LLAMA2-70B-CHAT performs notably better than its smaller counterpart, LLAMA2-13B-CHAT, showing a clear advantage due to larger model scale, though it still falls short of GPT-3.5-TURBO in most metrics, highlighting GPT-3.5-TURBO 's stronger coding capabilities. Interestingly, fine-tuned base LLMs perform particularly well in JavaScript, likely due to its simpler syntax and predictable patterns, which chat-LLMs like GPT-3.5-TURBO can easily understand and replicate, leading to high-quality training data. These findings highlight the effectiveness of "direct" responses and underscore the importance of data quality and style consistency in fine-tuning LLMs for code generation.

Data Curation Methods	HumanEval	MultiPL-E		
	Python Pass@1 / Pass@10	Java Pass@1 / Pass@10	JavaScript Pass@1 / Pass@10	C++ Pass@1 / Pass@10
Human Response	23.45 / 39.99	27.13 / 39.14	30.14 / 47.39	25.52 / 40.00
GPT-3.5-TURBO				
Referenced	28.58 / 52.64	29.46 / 41.91	33.53 / 50.84	27.70 / 42.17
Direct	33.00 / 51.48	29.38 / 42.03	33.19 / 51.72	28.45 / 43.27
LLAMA2-70B-CHAT				
Referenced	31.64 / 45.58	29.09 / 40.59	31.79 / 49.20	26.77 / 40.74
Direct	33.62 / 48.18	30.23 / 41.79	32.91 / 50.24	26.80 / 41.05
LLAMA2-13B-CHAT				
Referenced	23.88 / 43.31	27.58 / 37.92	29.90 / 47.72	25.09 / 38.67
Direct	28.32 / 40.99	24.67 / 36.41	28.88 / 45.65	23.81 / 36.96

Table 4: Detailed performance comparison of fine-tuned CODELLAMA-7B evaluated on HumanEval (Python) and MultiPL-E (Java, JavaScript, C++) coding benchmarks. The LLMs are fine-tuned on training sets curated with different response generation strategies and LLMs. The data examples are further filtered based on the perplexity similarity between "referenced" and human responses, excluding those with significant deviation. Pass@1 and Pass@10 scores for each programming language are reported.

B.2 Extended Stylometric Analysis

Evaluation Settings. To quantitatively evaluate stylistic consistency across datasets, we employ six stylometric metrics that capture distinct aspects of linguistic form, the structural elements that shape response presentation independent of semantics. Specifically, these metrics measure key linguistic form elements: transitional and functional word usage measured by TTR and MTLD of functional words, tone assessed by Flesch score, sentence structure quantified through Average Sentence Length, punctuation patterns captured by Punctuation Frequency, and layout features such as headers and bullet points measured by Layout Feature Frequency. Together with perplexity for assessing instructional surprisal, these metrics provide a comprehensive framework for analyzing response styles:
Linguistic Form Metrics:

1. **Type-Token Ratio (TTR):** Measures lexical diversity by calculating the ratio of unique words (types) to the total number of words (tokens) in a text. A higher TTR indicates greater lexical diversity.

2. **Measure of Textual Lexical Diversity (MTLD):** MTLD is less sensitive to text length compared to TTR. It computes the average length of sequential word strings that maintain a given TTR value, where higher MTLD scores suggest greater lexical diversity.

3. **Average Sentence Length (Avg. Sent. Len.):** Calculates the average number of words per sentence, providing insights into the syntactic complexity of the text.

4. **Punctuation Frequency (Punct. Freq.):** Computes the frequency of punctuation marks within each response, reflecting the density of punctuation usage.

5. **Flesch Reading Ease Score (Flesch Score):** Assesses readability based on the average sentence length and the average number of syllables per word. Higher scores indicate greater readability.

Data Curation Methods	TTR		MTLD		Avg. Sent. Len.		Punct. Freq.		Flesch Score		Avg. Layout Freq.		PPL$(y\|x)$	
	Mean	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
StackExchange														
Human Response	62.06	24.23	11.58	7.71	124.37	100.22	42.80	31.96	38.33	43.97	0.42	1.36	1.85	0.33
GPT-3.5-TURBO														
Referenced	31.65	8.16	13.61	2.51	46.49	20.90	44.88	25.38	57.32	16.16	0.10	0.28	1.84	0.33
Direct	34.15	8.14	13.34	2.57	46.31	23.59	38.80	20.48	54.66	16.92	0.26	0.41	1.78	0.30
LLAMA2-70B-CHAT														
Referenced	44.01	11.90	14.28	3.66	70.34	51.50	42.30	36.70	54.12	21.73	0.18	0.52	1.81	0.36
Direct	45.67	13.52	14.20	4.23	83.18	84.01	35.82	26.28	51.78	24.34	0.28	0.72	1.57	0.28
LLAMA2-13B-CHAT														
Referenced	31.97	7.46	15.64	3.06	43.03	25.11	50.31	28.81	62.73	17.23	0.13	0.42	1.76	0.27
Direct	33.35	8.86	14.90	3.12	43.49	27.49	39.60	22.64	61.44	16.92	0.22	0.38	1.76	0.28
LIMA														
Human Response	31.77	20.49	15.21	4.38	32.41	49.18	64.54	63.70	63.71	27.98	0.43	1.37	4.42	1.53
GPT-3.5-TURBO														
Referenced	48.40	18.43	15.28	6.04	26.51	21.36	14.27	10.73	59.45	19.25	0.15	0.64	4.02	1.52
Direct	47.53	16.06	15.08	5.31	24.87	17.04	14.08	9.33	55.59	21.00	0.26	0.58	2.51	0.64
LLAMA2-70B-CHAT														
Referenced	39.32	16.51	15.15	4.88	25.67	21.47	27.76	19.84	61.77	18.43	0.33	0.46	3.51	1.45
Direct	37.02	15.63	14.62	4.84	24.76	18.59	27.94	17.11	59.66	18.16	0.43	0.50	2.09	0.42
LLAMA2-13B-CHAT														
Referenced	35.74	13.64	15.98	4.42	24.65	14.75	27.44	17.70	64.46	17.45	0.16	0.42	3.10	1.19
Direct	31.90	14.22	15.08	3.78	22.60	12.61	35.22	18.74	62.30	15.40	0.37	0.39	2.06	0.38

Table 5: Comprehensive performance comparison of stylometric analysis across datasets using instructions from StackExchange and LIMA, paired with responses generated by human writers and various LLMs, presenting the average (Mean) and standard deviation (Std.) for six authorship detection metrics and Perplexity$(y|x)$.

6. **Layout Feature Frequency (Avg. Layout Freq.):** Calculates the frequency of structural elements (bullet points, headers, bold text) per sentence, representing the consistency of formatting and organizational patterns.

Instructional Surprisal Metric:

- **Perplexity of $P(y|x)$:** Captures the overall response surprisal given the instruction.

Discussion. Table 5 presents the average and standard deviation (*Std.*) of these metrics across responses from human-written and LLM-generated texts for both LIMA and StackExchange instructions. Our analysis reveals that LLM-generated responses consistently demonstrate higher stylistic consistency compared to human-written ones, with responses synthesized by GPT-3.5-TURBO and LLAMA2 showing lower standard deviations across most metrics. This indicates greater consistency in functional word diversity, sentence length, punctuation usage, readability, and layout features. Furthermore, "direct" responses achieve higher consistency in response surprisal than "referenced" and human responses, as evidenced by their lower standard deviation values of perplexities.

Notably, even the LIMA dataset, despite being optimized and curated by human experts for style consistency, exhibits lower stylistic consistency in our metrics compared to LLM-synthesized datasets. These results highlight both the inherent challenge of achieving style consistency through manual curation and the significant potential of using LLMs to generate stylistically consistent training data.

In conclusion, our stylometric analysis quantitatively validates that LLM-synthesized datasets demonstrate superior stylistic consistency compared to human-written responses across most measured dimensions.

B.3 Impact of Maintaining Instructional Surprisal Consistency in Referenced Responses on Stylometric Analysis and Model Performance

In Section 2, we applied perplexity-based filtering to remove instructions where "referenced" responses deviated significantly from human responses. Specifically, we excluded instructions where the PPL$(y|x)$ of at least one "Referenced" response exceeded thresholds of 0.15 or 2.5. This filtering process reduced the dataset to 944 instructions from StackExchange and 407 instructions from LIMA.

Data Curation Methods	StackExchange (10k)			LIMA (1k)		
	Stylometric Analysis	Data Quality	CodeLlama-7b Performance	Stylometric Analysis	Data Quality	Meta-Llama-3-8B Performance
	Std. TTR / Std. PPL	Helpfulness / Correctness	Avg. Pass@1 / Avg. Pass@10	Std. TTR / Std. PPL	Helpfulness / Correctness	L.C. WinRate
Human Response	22.27 / 1.41	3.34 / 3.57	31.65 / 46.63	19.54 / 8.01	4.32 / 4.37	2.29
GPT-3.5-turbo						
Referenced	7.95 / 0.31	3.65 / 3.60	31.66 / 48.82	17.43 / 5.86	4.05 / 4.32	4.07
Direct	7.75 / 0.28	3.55 / 3.50	35.11 / 49.68	16.43 / 3.61	4.18 / 4.49	7.15
Llama2-70b-chat						
Referenced	11.09 / 0.48	3.47 / 3.33	30.16 / 46.44	16.08 / 5.04	4.25 / 4.36	4.27
Direct	12.49 / 0.25	3.03 / 3.03	33.11 / 47.35	15.60 / 3.11	4.33 / 4.44	8.14
Llama2-13b-chat						
Referenced	7.29 / 0.24	2.82 / 2.54	26.88 / 42.87	12.96 / 3.49	4.03 / 4.00	3.94
Direct	8.27 / 0.22	2.09 / 1.93	25.13 / 37.73	13.18 / 1.13	3.66 / 3.78	6.80

Table 6: Performance comparison of CodeLlama-7b and Meta-Llama-3-8B fine-tuned on training sets curated using different methods and various LLMs, without applying surprisal-based instruction filtering, along with data quality and stylometric analysis metrics for the training sets.

Table 6 highlights the impact of dataset size on LLM fine-tuning performance in the coding domain. For human responses, the average Pass@1 score across all four programming languages increased from 26.56 to 31.65 after adding more data. Notably, the official base model CodeLlama-7b achieves a Pass@1 score of 29.98, while CodeLlama-7b-Instruct achieves 34.8 on HumanEval on BigCodeLeaderboard[3]. In contrast, Table 4 reports a significantly lower Pass@1 of 23.45, mainly due to the reduced dataset size (944 examples). **With sufficient data and effective selection strategies, the Pass@1 score on HumanEval for base CodeLlama-7b trained on human responses can reach 33, while synthetic responses can further boost performance to around 40, as shown in Tables 8 and 9.** As achieving high model performance is not the primary goal in Section 2, controlled filtering is essential for accurately analyzing variations in the instructional surprisal of responses and their impact on LLM fine-tuning.

A key observation from the stylometric analysis is the measurement of instructional surprisal through perplexity. Interestingly, Table 6 shows, without filtering, "referenced" responses exhibit greater surprisal consistency compared to human-written responses, particularly within the StackExchange code data. This finding is somewhat counterintuitive, as one might expect "referenced" responses–rewritten versions of human responses–to closely mirror the surprisal consistency of their human counterparts. We hypothesize that this discrepancy arises because LLMs, even when explicitly instructed to semantically align closely with human responses, may introduce subtle variations that affect surprisal metrics.

While perplexity-based filtering is critical for achieving a more accurate analysis of LLM performance under varying stylistic consistency conditions, it was not used for our SCAR training due to the following reasons: i) Table 6 shows that "Direct" responses already exhibit higher stylistic consistency than both "Referenced" and human responses, fulfilling the ranking objective. ii) Filtering removes a substantial number of examples, which could negatively impact training performance by reducing the dataset size.

B.4 Independence Tests of Linguistic Form and Instructional Surprisal

In this section, we examine whether the linguistic form features of responses are correlated with instructional surprisal and whether linguistic form depends on instructions. Understanding these relationships is essential for justifying the design of our ranking model, which employs distinct structures to represent these two sets of features.

Independence Between Linguistic Form and Instructional Surprisal. To validate the independence between linguistic form and instructional surprisal, we conduct two complementary analyses:

Regression Analysis: We perform regression modelling on the LIMA dataset to predict the instructional surprisal metric, perplexity $PPL(y|x)$, based on two feature sets:

[3] https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard

- **Linguistic form features:** unigrams of functional words, TTR and MTLD of functional words, punctuation and layout patterns, and Flesch readability scores.

- **Semantic features:** SentenceBERT (Reimers, 2019) embeddings derived from `sentence-transformers/all-MiniLM-L6-v2`, pre-trained for semantic encoding and paraphrase detection tasks.

The average absolute regression coefficients indicate that semantic features are significantly more influential in predicting instructional surprisal, with an average importance score of 1.193, compared to only 0.236 for each linguistic form feature.

Variance Analysis. We further investigate the independence of linguistic form and instructional surprisal by analyzing variance patterns in $PPL(y|x)$. Responses are decomposed into semantic tokens (y_c) and functional non-semantic tokens (y_p), which represent a key component of linguistic form elements (see Section B.6 for token separation details). By comparing the variance contributions of $PPL(y_c|y_p, x)$ and $PPL(y_p|y_c, x)$ to $PPL(y|x)$, we find:

- **Semantic tokens (y_c):** explain 283.67% of the variance.

- **Functional tokens (y_p):** explain only 4.01% of the variance.

The combined evidence from our regression and variance analyses confirms that linguistic form and instructional surprisal are independent dimensions of response style. Semantic features are the primary contributors to instructional surprisal, while linguistic form has a much weaker influence.

Independence Tests between Linguistic Form and Instructions We employ Conditional Mutual Information (CMI) (Wyner, 1978) to quantify the dependencies between semantic tokens (y_c) and non-semantic tokens (y_p) with respect to instructions (x). For semantic content and instructions, CMI is defined as:

$$I(y_c; x \mid y_p) = \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{P(y_c^{(i)} \mid x^{(i)}, y_p^{(i)})}{P(y_c^{(i)} \mid y_p^{(i)})} \right),$$

with an analogous formulation for functional tokens:

$$I(y_p; x \mid y_c) = \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{P(y_p^{(i)} \mid x^{(i)}, y_c^{(i)})}{P(y_p^{(i)} \mid y_c^{(i)})} \right).$$

Using META-LLAMA-3-8B to estimate conditional probabilities and a POS-based approach to separate semantic and non-semantic functional tokens (detailed in Appendix B.6), we analyze both human-written and GPT-3.5-TURBO-generated responses with LIMA and StackExchange instructions.

For LIMA instructions, the mutual information scores reveal that semantic tokens show a stronger dependence on instructions, with $I(y_c; x \mid y_p) = 0.4$, compared to $I(y_p; x \mid y_c) = 0.15$. Similarly, for StackExchange instructions, semantic tokens again dominate with $I(y_c; x \mid y_p) = 0.49$, while functional tokens exhibit a much weaker dependence at $I(y_p; x \mid y_c) = 0.03$. Since functional tokens are key indicators of linguistic form, these findings confirm that linguistic form has a significantly weaker dependence on instructions compared to semantic tokens. Therefore, in Eq. (3), we aim to use max pooling over their representations to capture linguistic form features as non-semantic surface characteristics of responses without explicitly modelling their relationship to the instruction. This approach aligns with our findings that linguistic form is largely independent of instructional context and has minimal influence on instructional surprisal.

B.5 Background on Surprisal Modeling

Text surprisal can be modelled in two primary ways:

Probability-based Surprisal. Surprisal of a word, traditionally defined as the negative log probability of a word given its context $-\log P(w \mid \text{context})$, has been widely recognized as a strong predictor of cognitive processing effort and neural responses, such as the N400 effect or other research (Oh and Schuler, 2023; Goodkind and Bicknell, 2018; Michaelov et al., 2023; Karampiperis et al., 2014). This measure can be directly computed using the next-token prediction objectives of language models. For example, the probability of a response given an instruction can theoretically be derived through the chain rule of probability:

$$P(\text{response} \mid \text{instruction}) = \prod_i P(w_i \mid w_1, \ldots, w_{i-1}, \text{instruction}),$$

where w_i is the i-th token of the response. While probabilistic models effectively capture word-level surprisal, transitioning to response-level surprisal introduces computational and conceptual complexities.

Representation-based Surprisal. Recent studies suggest that representation similarity can approximate surprisal by measuring semantic alignment between text spans (Michaelov et al., 2023; Karampiperis et al., 2014). This approach avoids direct probability computations, making it particularly suitable for capturing global semantic relationships between instructions and responses.

Challenges in Modeling Instructional Surprisal for SCAR. Prior work (Michaelov et al., 2023) employs simple word embeddings such as Word2Vec (Le and Mikolov, 2014) or GloVe (Pennington et al., 2014) and models surprisal as a scalar value through cosine similarity between text embeddings. While effective for certain semantic tasks, these methods face significant limitations in our context:

- **Limited Contextual Representations.** Word2Vec and GloVe embeddings, as employed in Michaelov et al. (2023), lack the contextual depth needed to capture the nuanced relationships between instructions and responses. Advanced encoders like RoBERTa or BERT offer significantly richer semantic representations, making them more suitable for modelling instructional surprisal within our framework and enhancing ranking performance.

- **Inadequacy of Linear Approaches.** Methods such as cosine similarity or Euclidean distances, as used in Michaelov et al. (2023); Karampiperis et al. (2014), rely on linear relationships to approximate mutual information between shallow text representations from Word2Vec or GloVe. While effective for these simpler embeddings, our preliminary experiments with advanced contextual embeddings from RoBERTa demonstrate that linear measures such as cosine or Euclidean distances between instruction and response embeddings fail to reveal meaningful patterns. This suggests that linear relationships may be sufficient for Word2Vec or GloVe but are inadequate for capturing the complex, non-linear relationships present in high-dimensional representations generated by models like BERT or RoBERTa. A non-linear approach is, therefore, essential to fully leverage these richer semantic representations.

- **Integration with Linguistic Form Modeling:** Our framework simultaneously models instructional surprisal and linguistic form, with linguistic features represented as high-dimensional distributed vectors. Reducing semantic alignment to a scalar value would create an imbalance in feature representation, limiting the model's ability to jointly optimize both dimensions. A distributed representation for surprisal is more compatible with our multi-dimensional modeling goals.

To overcome these limitations, we adopt a Relation Network as defined in Eq. (3) and leverage advanced contextual embeddings. This approach models instructional surprisal as a distributed representation, enabling the Relation Network to learn complex non-linear relationships while maintaining compatibility with the joint modelling of linguistic form.

Why Not Use Perplexity Directly? While perplexity is a traditional measure of surprisal, it is not well-suited for our task due to the following limitations:

- **Dependence on Large Models.** Accurate perplexity estimation requires large-scale pre-trained language models like META-LLAMA-3-8B, which are computationally expensive and misaligned with our goal of developing a scalable, efficient ranking-based data selection method.

- **Structural Misalignment.** Our framework models both linguistic form and instructional surprisal using distributed representations. Perplexity, as a scalar value, is inherently incompatible with this architecture, which relies on integrating distributed features for both aspects.

By modelling instructional surprisal as a distributed representation, our approach overcomes these challenges while maintaining compatibility with linguistic form modelling.

B.6 Identification of Semantic and Non-Semantic (Functional) Words

To distinguish between semantic content (y_c) and non-semantic (linguistic form-related) words (y_p) in the responses, we adopt a heuristic approach based on part-of-speech (POS) tagging. Specifically, content words–nouns, verbs, adjectives, and adverbs–are classified as semantic, while other POS tags (e.g., pronouns, conjunctions, prepositions, and determiners) are categorized as non-semantic.

For code-related responses, we also treat code blocks as semantic content, given their integral role in conveying the main content of the response. Code blocks are identified using regular expressions that capture common code delimiters, such as triple backticks (```), tildes (~~~), and inline code marked by single backticks (`).

Given the limitations of current NLP techniques, achieving perfect separation between semantic and non-semantic elements is challenging. However, our primary goal is not absolute precision, but to perform independence tests on various stylistic features relative to instructions and estimate instructional surprisal to inform our data selection ranker design. By focusing on comparative patterns, our approach effectively captures the impact of semantic and non-semantic content on stylistic consistency, and how these patterns influence data selection, ultimately improving LLM alignment through SFT.

To illustrate, Table 7 provides an example of how a response is split into semantic and non-semantic content using this method.

Instruction
Why is FIFA against adding instant replay to the game?
Response
FIFA is against adding instant replay to the game because they believe it would disrupt the flow of the game and potentially lead to delays. They also argue that human error is a part of the game and adding instant replay could take away from the excitement and unpredictability of live sports. Additionally, implementing instant replay would require significant changes to the rules and regulations of the game, which FIFA may be hesitant to do.

Table 7: Visualization of semantic and non-semantic words selected based on the POS tags in the response. Semantic words are in blue and functional words are in black.

C Extended Analysis of Main Experiments

C.1 Extended Analysis of Evaluation Results for Data Selection in Human-Written Coding Data

Table 8 offers a comprehensive breakdown of LLM performance when fine-tuned on datasets sampled using various data selection strategies, expanding upon the average results presented in Figure 2. While the figure provides aggregated metrics, this table delivers a detailed view of Pass@1 and Pass@10 scores for each programming language across the HumanEval and MultiPL-E benchmarks. This detailed presentation highlights performance variations in Python, Java, JavaScript, and C++.

The performance ranking of data selection methods aligns consistently with the trends shown in Figure 2, reinforcing our findings' reliability. Strategies such as SCAR(ID) and Perplexity-based sampling demonstrate robust performance across most languages, while approaches like HFR and Superfiltering

Data Sampling Methods	HumanEval	MultiPL-E		
	Python Pass@1 / Pass@10	Java Pass@1 / Pass@10	JavaScript Pass@1 / Pass@10	C++ Pass@1 / Pass@10
Full Data	32.87 / 48.24	30.92 / 44.92	33.84 / 52.62	28.51 / 43.91
SCAR (OOD)				
50%	31.94 / 47.80	30.85 / 43.29	33.91 / 52.45	29.23 / 45.28
25%	31.85 / 46.80	29.97 / 43.24	33.14 / 52.75	29.20 / 45.21
12.5%	30.77 / 46.80	28.92 / 41.86	31.23 / 48.38	28.17 / 43.61
SCAR (ID)				
50%	33.83 / 50.24	30.10 / 44.95	34.46 / 53.10	28.25 / 43.71
25%	31.48 / 48.68	30.76 / 44.60	32.91 / 52.15	28.92 / 43.98
12.5%	31.10 / 47.14	29.46 / 43.06	31.38 / 49.11	27.61 / 42.39
Random				
50%	29.79 / 44.06	30.14 / 43.90	32.86 / 51.61	28.48 / 43.89
25%	30.04 / 45.76	30.22 / 42.35	33.06 / 51.05	28.89 / 43.89
12.5%	27.94 / 45.79	27.53 / 40.47	31.48 / 51.25	25.29 / 40.51
Perplexity				
50%	33.27 / 47.90	29.73 / 42.16	32.67 / 52.13	28.46 / 43.40
25%	32.29 / 47.05	29.33 / 42.40	32.45 / 50.10	28.73 / 44.78
12.5%	27.40 / 45.13	28.67 / 40.77	31.30 / 50.71	26.36 / 41.75
Superfiltering				
50%	26.50 / 42.00	29.72 / 43.53	32.97 / 52.40	27.86 / 44.86
25%	24.12 / 38.51	29.29 / 42.76	32.50 / 53.20	26.89 / 41.01
12.5%	8.22 / 25.58	26.79 / 38.83	30.11 / 49.20	23.99 / 36.82
HFR				
50%	20.29 / 41.52	30.41 / 44.11	33.49 / 51.27	28.71 / 44.83
25%	11.20 / 25.73	29.38 / 42.81	31.73 / 51.51	28.09 / 43.07
12.5%	11.04 / 27.74	27.51 / 40.82	30.71 / 49.41	24.91 / 39.77
AlpaGasus				
50%	31.30 / 44.90	30.59 / 43.41	34.21 / 52.48	29.45 / 43.91
25%	30.32 / 45.00	29.73 / 42.78	32.24 / 51.65	28.29 / 44.15
12.5%	24.76 / 41.90	28.24 / 42.12	30.84 / 49.56	26.17 / 41.12
Diversity				
50%	33.05 / 48.38	30.53 / 44.06	34.02 / 53.99	28.84 / 42.60
25%	30.38 / 44.52	30.04 / 42.53	33.34 / 52.71	28.68 / 44.66
12.5%	25.87 / 44.07	27.35 / 39.37	30.48 / 49.65	24.99 / 40.38
Longest				
50%	30.99 / 50.90	30.74 / 44.74	32.17 / 52.47	28.32 / 43.55
25%	30.10 / 48.41	29.35 / 42.65	30.72 / 51.98	28.92 / 45.07
12.5%	28.12 / 47.60	28.54 / 41.97	29.53 / 48.43	27.40 / 41.65

Table 8: Detailed performance comparison of fine-tuned CODELLAMA-7B evaluated on the HumanEval (Python) and MultiPL-E (Java, JavaScript, C++) coding benchmarks. The models are fine-tuned on human-written datasets selected with different selection methods and proportions. The table reports Pass@1 and Pass@10 scores for each individual programming language.

yield less favourable results, particularly with smaller data proportions. Notably, LLMs trained on our SCAR(ID)-selected data outperform those trained on the full dataset when the selection portion exceeds 25%, highlighting the superiority of our method. This result indicates that a carefully curated subset can sometimes produce better outcomes than using the entire dataset.

For a detailed explanation of the Pass@1 and Pass@10 metrics, please refer to the HumanEval paper by Chen et al. (2021).

Data Sampling Methods	HumanEval Python Pass@1 / Pass@10	MultiPL-E Java Pass@1 / Pass@10	MultiPL-E JavaScript Pass@1 / Pass@10	MultiPL-E C++ Pass@1 / Pass@10
Full Data	40.63 / 54.93	32.67 / 44.24	36.89 / 54.10	32.68 / 45.65
SCAR (OOD)				
50%	40.15 / 55.25	32.15 / 44.44	37.01 / 55.59	31.96 / 46.59
25%	38.23 / 52.58	32.57 / 45.44	37.04 / 53.20	30.60 / 45.67
12.5%	38.29 / 52.74	32.46 / 45.45	36.07 / 53.45	31.91 / 45.56
SCAR (ID)				
50%	40.98 / 56.57	32.80 / 45.75	37.58 / 55.69	32.73 / 45.71
25%	39.84 / 56.75	32.52 / 43.83	36.67 / 55.32	32.00 / 46.26
12.5%	36.93 / 52.96	32.62 / 44.82	36.45 / 52.33	30.43 / 45.42
Random				
50%	39.04 / 51.80	31.75 / 44.85	35.59 / 55.13	32.76 / 46.34
25%	35.61 / 52.40	31.33 / 44.24	36.68 / 54.23	30.53 / 44.60
12.5%	34.99 / 51.90	31.34 / 44.29	35.91 / 51.63	31.08 / 44.49
Perplexity				
50%	31.91 / 50.94	32.44 / 45.37	37.02 / 54.75	33.22 / 46.19
25%	35.55 / 48.65	31.85 / 45.44	35.40 / 51.75	31.28 / 43.32
12.5%	27.37 / 43.06	30.90 / 44.19	36.34 / 48.74	30.46 / 42.96
Superfiltering				
50%	38.93 / 54.55	31.80 / 44.48	35.03 / 54.40	32.22 / 47.25
25%	35.93 / 51.41	32.47 / 44.10	34.46 / 53.13	30.89 / 44.90
12.5%	34.35 / 49.81	30.34 / 42.81	32.97 / 50.60	30.46 / 44.22
HFR				
50%	39.09 / 53.59	32.42 / 43.90	36.11 / 53.51	31.60 / 45.51
25%	38.04 / 53.36	32.57 / 43.51	36.45 / 54.10	31.27 / 46.28
12.5%	29.20 / 50.06	31.87 / 43.85	35.17 / 53.94	30.02 / 44.31
AlpaGasus				
50%	36.88 / 53.05	32.20 / 45.65	36.57 / 54.84	33.07 / 45.77
25%	32.52 / 49.55	31.37 / 42.82	33.32 / 51.72	30.37 / 44.69
12.5%	29.08 / 45.07	31.09 / 43.09	34.82 / 52.53	29.73 / 44.16
Diversity				
50%	39.21 / 54.95	32.10 / 45.48	37.25 / 54.58	32.60 / 46.33
25%	35.29 / 51.33	32.00 / 43.41	36.10 / 55.44	30.98 / 45.19
12.5%	33.60 / 50.18	31.78 / 44.92	34.82 / 51.92	30.91 / 44.10
Longest				
50%	36.83 / 53.90	32.73 / 45.15	36.73 / 55.92	33.85 / 46.83
25%	35.60 / 53.50	32.34 / 45.54	36.25 / 54.65	32.57 / 46.43
12.5%	34.54 / 49.89	32.41 / 46.31	35.57 / 54.64	31.42 / 45.30

Table 9: Detailed performance comparison of fine-tuned CODELLAMA-7B evaluated on the HumanEval (Python) and MultiPL-E (Java, JavaScript, C++) coding benchmarks. The models are all fine-tuned using GPT-3.5-TURBO-generated datasets selected with different data selection methods and varying proportions. The table reports the Pass@1 and Pass@10 scores for each individual programming language.

C.2 Extended Analysis of Evaluation Results for Data Selection in Mixed Synthetic Coding Data

Table 9 offers a detailed breakdown of the LLM performance results summarized in Figure 2. It presents Pass@1 and Pass@10 scores across four programming languages, evaluating LLMs fine-tuned on synthetic dataset subsets chosen through various selection methods. This comprehensive view provides insights into the LLM's performance on individual tasks and programming languages, complementing the aggregated results shown in the figure.

C.3 Extended Analysis of Evaluation Results for Open-Domain Data Selection Experiments

	Methods								
	SCAR (ID)	**SCAR (OOD)**	**Random**	**Perplexity**	**Superfiltering**	**HFR**	**AlpaGasus**	**Diversity**	**Longest**
Human									
100%					2.34				
50%	2.24	1.90	2.03	1.74	2.00	1.50	2.09	1.99	1.46
25%	2.43	2.59	1.92	2.12	1.82	1.66	1.83	1.97	1.75
10%	2.67	2.02	2.13	2.51	2.04	2.21	1.96	2.03	1.27
Synthetic									
100%					3.64				
50%	5.56	5.31	2.61	4.17	4.22	3.86	3.86	3.56	6.29
25%	5.89	5.08	3.00	4.04	5.70	4.30	3.94	2.51	5.32
10%	6.61	4.94	2.38	4.54	5.38	4.06	4.78	3.02	6.61

Table 10: Detailed comparison of Length Control WinRate for fine-tuned META-LLAMA-3-8B models evaluated on AlpacaEval benchmarks. Models are trained using human-written and synthetic GPT-3.5-TURBO-generated data, sampled with various selection methods and proportions.

Table 10 presents the detailed numerical values for the Length Control WinRate, complementing the visual representation provided in Figure 2. The results show that for the selection of human data, SCAR(ID) and SCAR(OOD) achieve competitive performance even at reduced data proportions, with SCAR(ID) showing a slight advantage as the data size decreases, especially at the 25% and 10% subsets. In contrast, methods such as Random and HFR struggle to maintain consistently high performance across different data scales.

For the selection of synthetic GPT-3.5-TURBO-generated data, SCAR(ID) consistently outperforms all methods except Longest, with WinRates peaking at 6.61 for the 10% subset. Interestingly, Longest performs comparably to SCAR(ID) when selecting synthetic data, as it tends to favour Evol-Instruct-generated data, which produces longer responses. This finding highlights that response token length can serve as a strong stylistic indicator, aligning with the principles of our style consistency framework.

These results suggest that well-curated synthetic datasets can enable high-performing chat-LLMs even at significantly reduced data proportions. Furthermore, traditional methods such as Random and Perplexity exhibit lower performance, underscoring the importance of selection strategies tailored to stylistic consistency in synthetic data scenarios. Striking a balance between data size, diversity, and style consistency remains crucial for optimizing performance.

C.4 Extended Analysis of Style and Quality Analysis in SCAR-Selected Data

Table 11 presents an extensive set of results, expanding upon the data shown in Table 2. In addition to helpfulness and correctness scores, as well as the standard deviations of TTR and perplexity, this table includes a comprehensive range of stylometric and quality metrics with their corresponding average and standard deviation values. The results are consistent with our findings in Table 2. SCAR selection effectively enhances the consistency of the linguistic form in the selected data, as evidenced by the consistently decreasing standard deviation values across **most** linguistic form metrics as the selection portion decreases. Similarly, the standard deviation of instructional surprisal metrics generally decreases, except in a few cases when selecting smaller portions (e.g., 25%, 12.5%) of human-written or synthetic code data.

Interestingly, while the standard deviations of TTR and MTLD for functional words decrease, their mean values remain largely unaffected–and in some cases, even increase. This suggests that SCAR selection preserves the overall lexical diversity of functional words while narrowing their variability across examples, resulting in more consistent usage. In other words, the coverage of functional word choices is maintained (as reflected by stable or higher mean values); however, SCAR's ranking mechanism enhances response stylistic consistency by reducing outliers and extreme variations of linguistic forms, leading to lower standard deviations. This indicates that SCAR does not inherently restrict lexical diversity in linguistic form; rather, it ensures that linguistic form features are applied more uniformly throughout the

	TTR		MTLD		Avg. Sent. Len.		Punct. Freq.		Flesch Score		Avg. Layout Freq.		PPL$(y \mid x)$		Helpful	Correct
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.		
Code Domain																
Human																
100%	59.16	21.48	15.05	8.37	69.40	66.43	30.77	27.17	42.75	44.36	0.25	0.81	3.83	1.81	2.84	2.68
50%	50.80	16.78	16.34	6.30	68.16	65.49	37.23	28.53	48.59	30.68	0.21	0.67	3.77	1.72	3.02	3.01
25%	47.43	14.85	16.58	5.28	53.36	48.11	34.93	27.10	49.84	24.60	0.20	0.63	3.84	1.73	2.78	2.72
12.5%	45.78	14.29	16.45	4.98	50.50	49.46	33.35	25.42	51.26	22.25	0.20	0.54	3.93	1.86	2.67	2.77
Synthetic																
100%	36.67	14.45	12.13	3.87	60.88	61.39	37.72	24.62	49.17	23.10	0.10	0.49	1.67	0.31	3.63	3.64
50%	36.79	10.52	13.07	2.80	52.85	36.48	35.49	22.01	50.52	16.87	0.14	0.63	1.74	0.31	3.52	3.56
25%	36.67	9.33	13.29	2.75	48.71	27.26	31.70	17.62	51.19	15.94	0.21	0.85	1.83	0.34	3.47	3.44
12.5%	37.19	9.22	13.52	2.98	48.36	28.54	28.93	17.02	51.42	16.03	0.25	0.45	1.94	0.35	3.55	3.39
Open Domain																
Human																
100%	54.51	30.96	8.93	8.00	19.90	16.66	7.62	12.22	61.21	28.03	0.25	1.42	5.23	3.26	3.95	3.91
50%	61.24	28.43	9.55	7.92	21.35	16.36	6.58	8.84	58.27	24.33	0.34	1.76	4.57	2.69	3.98	3.99
25%	62.81	24.74	18.58	7.52	23.49	17.22	6.92	9.32	55.54	21.76	0.40	2.03	4.17	2.41	3.96	3.93
10%	57.01	23.73	11.26	6.77	25.44	20.01	7.71	7.16	51.78	22.40	0.60	2.71	3.93	2.18	3.98	3.99
Synthetic																
100%	55.15	30.04	9.87	7.67	23.76	32.82	12.30	20.53	54.40	71.06	0.29	1.27	2.75	1.16	3.93	3.96
50%	47.78	21.08	13.30	5.71	27.33	25.25	18.12	22.09	48.61	21.62	0.35	1.17	2.38	0.72	3.99	3.99
25%	41.96	17.34	13.83	4.40	24.59	18.42	20.54	19.19	46.47	19.89	0.41	1.14	2.33	0.61	3.98	4.02
10%	40.53	14.83	14.15	3.87	21.49	11.93	20.99	15.92	42.04	17.74	0.39	0.80	2.46	0.52	4.00	4.02

Table 11: Detailed performance comparison of the stylometric analysis conducted across the full datasets and the subsets of the full datasets selected by SCAR(ID) in both code and open domains. The table reports the average and standard deviation for six authorship metrics, perplexity, and average helpfulness and correctness scores.

	SCAR(ID)		SCAR(OOD)	
	Code	Open	Code	Open
$\text{Acc}(y^d \succ y^r \succ y^h)$	98.20	64.77	64.26	45.85
$\text{Acc}(y^d \succ y^r)$	98.40	80.80	68.29	67.88
$\text{Acc}(y^r \succ y^h)$	99.80	81.47	95.58	69.89

Table 12: SCAR's ranking accuracies when trained with in-domain or out-of-domain examples and tested on ranking data from code and open domains.

dataset.

C.5 Analysis of Ranker Performance

Evaluation Settings. We report the accuracy of the ranker in correctly rating responses on the test, where the goal is to rate "direct" responses higher than "referenced" responses and "referenced" responses higher than human responses. These accuracies are denoted as $\text{Acc}(y^d \succ y^r \succ y^h)$, $\text{Acc}(y^r \succ y^h)$, and $\text{Acc}(y^d \succ y^r)$, respectively.

Impact of SCAR Performance. Table 12 shows accuracies of SCAR(OOD) are lower than SCAR(ID) in both domains, explaining the lower LLM performance with SCAR(OOD)-selected data. Despite this, SCAR(OOD) outperforms selection baselines in most cases, demonstrating its cross-domain robustness. The ranking accuracy gap between SCAR(OOD) and SCAR(ID) is larger in the open domain, indicating that generalizing from code to open-ended data is more challenging than the reverse. Differentiating surprisal-related features is more difficult than differentiating linguistic form, especially for selecting code data in out-of-domain settings, as shown by comparing $\text{Acc}(y^d \succ y^r)$ (68.29) and $\text{Acc}(y^r \succ y^h)$ (95.58).

C.6 Extended Evaluation Analysis of STARCODER-15.5B

Table 13 presents the full Pass@1 and Pass@10 results for the HumanEval and MultiPL-E coding benchmarks, comparing STARCODER-15.5B fine-tuned with various portions of SCAR-selected data against OCTOCODER-15.5B. The original dataset, comprising 13k examples, was curated by the BigCode team, who developed both STARCODER-15.5B and OCTOCODER-15.5B and fine-tuned STARCODER-15.5B into OCTOCODER-15.5B. Notably, STARCODER-15.5B models fine-tuned on SCAR-selected subsets outperform the original OCTOCODER-15.5B in Pass@1 and Pass@10 across all programming languages.

The Pass@1 score of OCTOCODER-15.5B for HumanEval-Python on the BigCode leaderboard is 45.3, which corresponds to the `humanevalsynthesize-python` benchmark. This variant of `humaneval-python` employs improved prompt formatting, resulting in higher performance. In contrast, our paper reports OCTOCODER-15.5B's Pass@1 score of 35.56 on the standard `humaneval-python` benchmark to maintain consistency with widely accepted evaluation protocols and the default settings used in our experiments. Both results are sourced from the official BigCode leaderboard data files[4]. For further details, please refer to the provided data file URL and the benchmark description in Muennighoff et al. (2023) to understand the design differences between `humanevalsynthesize-python` and `humaneval-python`.

Data Sampling Methods	HumanEval	MultiPL-E		
	Python	Java	JavaScript	C++
	Pass@1 / Pass@10	Pass@1 / Pass@10	Pass@1 / Pass@10	Pass@1 / Pass@10
OCTOCODER-15.5B	35.56 / 51.81	26.03 / 38.44	32.80 / 46.97	29.32 / 41.90
STARCODER-15.5B				
10,000	36.29 / 53.99	28.29 / 39.58	33.22 / 49.79	30.17 / 46.20
5,000	36.95 / 54.07	28.96 / 39.02	34.53 / 49.90	32.83 / 44.47
2,500	37.57 / 55.65	29.29 / 41.06	34.09 / 49.47	31.19 / 42.83

Table 13: Detailed performance comparison of OCTOCODER-15.5B and STARCODER-15.5B fine-tuned on various subsets of the 13k data used to train OCTOCODER-15.5B. The models are evaluated on the HumanEval (Python) and MultiPL-E (Java, JavaScript, C++) coding benchmarks.

C.7 Extended Evaluation of Data Selection Performance for LLMs on Four Additional Benchmarks: ARC-Challenge, HellaSwag, MMLU, and TruthfulQA

Model Variants	Data Size	ARC-Challenge ACC (LHH)	HellaSwag ACC (LHH)	MMLU ACC (SM)	TruthfulQA BLEU	AlpacaEval L.C. WinRate	Average Rank↓
OLMO-7B (allenai/tulu-v2-sft-mixture)	320k	39.42	75.06	**38.60**	33.90	3.86	3.2
	10k	**41.04**	75.18	25.40	38.31	5.37	2.6
	5k	39.08	**75.33**	26.28	40.02	**5.64**	2.2
	2.5k	39.76	75.29	26.41	**40.39**	4.08	**2.0**
META-LLAMA-3-8B (Mixed Synthetic Data)	10k	**55.72**	79.02	40.04	19.34	3.64	3.4
	5k	50.85	79.06	54.45	37.21	5.56	2.7
	2.5k	49.40	**79.31**	**54.60**	37.58	5.89	2.0
	1k	51.88	79.06	48.79	**39.90**	**6.61**	**1.9**
META-LLAMA-3-8B (Human-written Data)	10k	53.41	**81.07**	34.02	33.90	2.34	2.6
	5k	**55.46**	80.56	28.28	34.52	2.24	2.8
	2.5k	54.35	80.22	31.13	34.88	2.43	2.4
	1k	47.35	80.15	**35.62**	**37.09**	**2.67**	**2.2**

Table 14: Performance comparison on five benchmarks: ARC-Challenge (Accuracy calculated with Likelihood), HellaSwag (Accuracy calculated with Likelihood), MMLU (Accuracy using String Matching), TruthfulQA (BLEU comparison), AlpacaEval (L.C. WinRate), and Average Rank. The table includes fine-tuned versions of OLMO-7B on human-written data and META-LLAMA-3-8B fine-tuned on mixed synthetic and human-written data across varying dataset sizes (320k, 10k, 5k, 2.5k, and 1k).

Evaluation Settings. Table 14 provides a detailed evaluation of fine-tuned OLMO-7B and META-LLAMA-3-8B models across five diverse benchmarks: ARC-Challenge, TruthfulQA, HellaSwag, MMLU, and AlpacaEval. These benchmarks includes a wide range of tasks, from general knowledge and reasoning to language understanding and text generation, offering a comprehensive assessment of LLM SFT performance.

- **ARC-Challenge** (Clark et al., 2018): Evaluates multiple-choice science questions, with accuracy measured using Likelihood (LHH).

[4] https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard/tree/main/community_results/bigcode_octocoder_loubnabnl/metrics_octocoder

- **TruthfulQA** (Lin et al., 2022): Measures factual precision and correctness in LLM responses using BLEU scores.
- **HellaSwag** (Zellers et al., 2019): Tests common-sense reasoning and contextual understanding, with accuracy calculated using Likelihood (LHH).
- **MMLU** (Hendrycks et al.): Focuses on multi-task language understanding, with accuracy evaluated using String Matching (SM).
- **AlpacaEval**: Assesses open-domain instruction-following capabilities, using Length Control Win-Rate (L.C. WinRate) as the metric.

Additionally, an **average ranking metric** is used to aggregate performance across benchmarks, with lower ranks indicating better overall performance. The average ranking is chosen instead of average performance because it balances variations across metrics, preventing benchmarks with different scales (e.g., BLEU and accuracy) from disproportionately influencing the results.

For TruthfulQA and MMLU, String Matching and BLEU scores are used instead of Likelihood-based metrics to better align with the nature of instruction-tuned models, which are optimized for generating complete answers rather than reproducing ground truth tokens. However, as we rely on `lm-evaluation-harness`[5], it lacks direct support for implementing these metrics for ARC-Challenge and TruthfulQA, constraining us to use Likelihood for these benchmarks.

Discussion. Table 14 demonstrates that subsets selected by SCAR(ID) from larger datasets can consistently outperform models trained on full data in most cases, aligning with our findings in Table 3 in the main body of the paper. Notably, subsets selected using our SCAR method show substantial performance improvements. For example, OLMO-7B fine-tuned on a SCAR-selected subset (e.g., 2.5k examples) achieves superior average rankings compared to the 320k full dataset on benchmarks like TruthfulQA (BLEU: 40.39 vs. 33.90) and AlpacaEval (L.C. WinRate: 4.08 vs. 3.86). Similarly, META-LLAMA-3-8B fine-tuned on a 2.5k subset of mixed synthetic data curated with SCAR outperforms larger subsets on MMLU (Accuracy: 54.60) and AlpacaEval (L.C. WinRate: 5.89), achieving a top average rank of 2.0.

These results highlight the effectiveness of our SCAR selection method in optimizing fine-tuned LLM performance across diverse benchmarks. By prioritizing data quality and style consistency, SCAR-selected subsets not only reduce computational costs but also enhance model generalization.

D Extended Analysis of Ablation Studies

Tables 15 and 16 present detailed performance metrics for various CODELLAMA-7B-based models. These models were fine-tuned on different data subsets selected by SCAR from full datasets with either human-written or synthetic responses, with instructions derived from StackExchange. The tables illustrate the performance of fine-tuned LLMs when using SCAR with various components removed during SCAR training. This comparison allows us to assess the impact of each SCAR component on the LLM fine-tuning performance. Unlike the summary results in Figure 3, these tables offer specific numerical values, enabling clearer and more precise comparisons. The results demonstrate that removing almost any component of SCAR during ranker training reduces LLM fine-tuning performance, regardless of whether the data is sourced from human or synthetic origins in the coding domain. This finding validates the importance of each element in our ranker design.

To further explore the impact of representation learning (w/o rl, GPT-3.5) and "referenced" responses (w/o ref, GPT-3.5) during SCAR training, we conducted two additional analyses, which are detailed in the following sections.

D.1 Impact of Training SCAR without Referenced Responses

As shown in Table 17, excluding "referenced" responses during SCAR(ID) training significantly reduces the performance of META-LLAMA-3-8B fine-tuned on SCAR-selected open-domain data subsets when

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

Data Sampling Methods	HumanEval	MultiPL-E		
	Python	Java	JavaScript	C++
	Pass@1 / Pass@10	Pass@1 / Pass@10	Pass@1 / Pass@10	Pass@1 / Pass@10
	Human Data			
Full, GPT-3.5				
50%	32.44 / 50.38	30.67 / 44.86	34.40 / 53.16	29.49 / 45.73
25%	31.98 / 49.25	30.41 / 43.65	34.04 / 52.72	29.19 / 43.41
12.5%	31.10 / 47.14	29.46 / 43.06	31.38 / 49.11	27.61 / 42.39
w/o con, GPT-3.5				
50%	31.21 / 50.01	30.14 / 44.23	34.67 / 51.90	28.67 / 43.90
25%	31.19 / 47.83	31.22 / 45.73	32.91 / 52.41	28.32 / 44.85
12.5%	30.13 / 45.39	28.72 / 42.68	30.99 / 49.60	27.39 / 42.85
w/o rl, GPT-3.5				
50%	33.60 / 50.02	30.47 / 44.53	33.88 / 52.96	28.91 / 45.22
25%	31.76 / 47.47	30.73 / 43.98	32.51 / 51.11	29.42 / 43.47
12.5%	30.56 / 45.26	28.82 / 43.19	31.24 / 49.35	26.89 / 40.95
w/o ref, GPT-3.5				
50%	33.63 / 49.22	31.06 / 45.11	34.45 / 53.41	28.66 / 43.96
25%	31.57 / 48.06	30.84 / 44.26	32.89 / 52.58	29.24 / 45.05
12.5%	30.62 / 45.98	28.06 / 40.71	30.80 / 48.08	28.16 / 42.80
Full, Llama2-70b				
50%	33.27 / 49.42	30.49 / 43.21	33.70 / 51.46	29.24 / 44.27
25%	29.47 / 46.12	29.75 / 43.19	33.33 / 49.69	29.17 / 44.39
12.5%	30.76 / 46.79	28.13 / 40.52	31.23 / 50.34	27.66 / 41.58
Full, Llama2-13b				
50%	31.90 / 50.38	30.75 / 44.29	33.34 / 51.81	28.62 / 42.57
25%	31.71 / 48.49	29.78 / 43.73	32.20 / 51.25	28.40 / 43.16
12.5%	30.29 / 46.03	28.18 / 42.03	30.70 / 48.19	27.47 / 41.58
w/o con, Llama2-13b				
50%	30.76 / 43.63	29.84 / 44.11	32.07 / 51.50	28.04 / 43.07
25%	30.15 / 42.78	29.44 / 43.66	32.88 / 54.14	27.93 / 44.26
12.5%	27.93 / 41.07	27.28 / 39.27	31.18 / 49.99	25.57 / 41.35
Full, Llama3-70b				
50%	32.48 / 50.39	30.68 / 45.30	33.49 / 53.01	29.28 / 45.13
25%	32.28 / 49.14	30.04 / 43.86	32.09 / 51.54	28.09 / 43.63
12.5%	30.40 / 48.36	28.14 / 41.71	30.67 / 49.67	26.99 / 42.47

Table 15: Comprehensive performance comparison of CODELLAMA-7B models fine-tuned on human-written datasets, evaluated on HumanEval (Python) and MultiPL-E (Java, JavaScript, C++) coding benchmarks. The training datasets were sampled using various methods at different proportions. Pass@1 and Pass@10 scores are reported for each programming language.

evaluated on the AlpacaEval benchmark. This result underscores the importance of incorporating "referenced" responses during ranker training to ensure the ranker effectively captures representations that model the instructional surprisal of responses in the open domain. In the code domain, however, excluding "referenced" responses during SCAR training has only a minor effect on data selection and LLM SFT performance.

D.2 Representation Similarities Analysis

As shown in Table 18, we calculate the cosine similarities between linguistic form representations ($\mathbf{v}_p$) and instructional surprisal representations ($\mathbf{v}_c$) for "direct", "referenced", and human-written responses. Specifically, the table reports the cosine similarities between i) "direct" and "referenced" responses, ii) "referenced" and human-written responses, and iii) "direct" and human-written responses for both linguistic form and instructional surprisal representations. According to Eq. 4, we expect the similarity between "direct" and "referenced" responses to be higher than those between "referenced" and human or

Data Sampling Methods	HumanEval	MultiPL-E		
	Python Pass@1 / Pass@10	Java Pass@1 / Pass@10	JavaScript Pass@1 / Pass@10	C++ Pass@1 / Pass@10
Mixed Synthetic Data				
Full, GPT-3.5				
50%	40.98 / 56.57	32.80 / 45.75	37.58 / 55.69	32.73 / 45.71
25%	39.84 / 56.75	32.52 / 43.83	36.67 / 55.32	32.00 / 46.26
12.5%	36.93 / 52.96	32.62 / 44.82	36.45 / 52.33	30.43 / 45.42
w/o con, GPT-3.5				
50%	39.65 / 55.05	32.30 / 44.40	38.21 / 54.92	32.17 / 45.66
25%	39.30 / 56.87	32.76 / 45.87	37.43 / 54.76	32.11 / 45.77
12.5%	36.56 / 51.72	33.00 / 44.48	35.53 / 53.10	31.02 / 45.44
w/o rl, GPT-3.5				
50%	39.83 / 54.27	32.28 / 43.66	37.66 / 55.99	32.53 / 46.31
25%	38.62 / 56.03	32.55 / 43.67	36.75 / 53.65	32.25 / 45.06
12.5%	36.02 / 51.78	32.71 / 45.68	35.70 / 52.15	31.70 / 45.51
w/o ref, GPT-3.5				
50%	39.85 / 55.81	32.13 / 44.00	36.87 / 56.79	32.67 / 46.43
25%	36.80 / 54.70	32.68 / 45.91	36.87 / 57.04	31.61 / 47.02
12.5%	36.41 / 50.96	32.66 / 44.58	35.78 / 52.21	30.99 / 44.88
Full, Llama2-70b				
50%	39.21 / 52.49	32.39 / 45.21	37.45 / 54.87	33.03 / 46.36
25%	39.23 / 53.77	31.59 / 45.21	37.35 / 55.15	30.81 / 45.04
12.5%	37.59 / 51.64	31.44 / 44.82	37.04 / 52.55	30.67 / 44.80
Full, Llama2-13b				
50%	37.29 / 53.60	33.24 / 43.86	37.04 / 56.29	32.36 / 44.65
25%	36.70 / 51.88	31.97 / 44.57	36.35 / 56.33	31.12 / 46.04
12.5%	33.78 / 48.61	30.61 / 41.77	34.21 / 51.66	31.11 / 45.27
w/o con, Llama2-13b				
50%	37.72 / 53.82	32.18 / 44.19	37.23 / 56.76	32.57 / 46.31
25%	38.59 / 53.47	32.68 / 44.97	37.19 / 55.59	32.00 / 46.58
12.5%	33.34 / 49.78	32.05 / 43.76	35.58 / 53.38	31.02 / 46.13
Full, Llama3-70b				
50%	39.40 / 54.46	32.87 / 45.00	36.99 / 57.26	32.52 / 46.38
25%	38.40 / 54.73	32.54 / 44.79	37.40 / 54.46	30.92 / 44.06
12.5%	35.48 / 50.33	31.80 / 45.40	36.45 / 53.71	30.99 / 46.66

Table 16: Comprehensive performance comparison of CODELLAMA-7B models fine-tuned on GPT-3.5-TURBO-generated datasets, evaluated on HumanEval (Python) and MultiPL-E (Java, JavaScript, C++) coding benchmarks. The training datasets were selected from the full mixed synthetic dataset with different sample sizes using our selection approach, SCAR(ID), with various training configurations. Pass@1 and Pass@10 scores are reported for each programming language.

	Human			Mix Synthetic		
	50%	25%	10%	50%	25%	10%
Full	2.24	2.43	2.67	5.56	5.89	6.61
w/o ref	1.95	2.25	1.99	3.59	4.74	4.44

Table 17: Comparison of L.C. WinRate on the AlpacaEval benchmark for META-LLAMA-3-8B fine-tuned on subsets of human-written and synthetic data selected by SCAR(ID), with and without incorporating "referenced" responses during ranker training.

"direct" and human responses for linguistic form representations. Conversely, for instructional surprisal representations, the similarity between "referenced" and human responses should be the highest.

Interestingly, even without the representation learning regularization loss in Eq. 4 and while incorporating "referenced" responses during SCAR training, the observed cosine similarities still align with

	Linguistic Form Representation			Instructional Surprisal Representation		
	$\cos(\mathbf{v}_p^d, \mathbf{v}_p^r)$	$\cos(\mathbf{v}_p^r, \mathbf{v}_p^h)$	$\cos(\mathbf{v}_p^d, \mathbf{v}_p^h)$	$\cos(\mathbf{v}_c^d, \mathbf{v}_c^r)$	$\cos(\mathbf{v}_c^r, \mathbf{v}_c^h)$	$\cos(\mathbf{v}_c^d, \mathbf{v}_c^h)$
	LIMA					
SCAR(ID)	0.9368	0.8970	0.7884	0.8312	0.8801	0.7209
SCAR(ID) w/o rl	0.9050	0.7962	0.6369	0.9406	0.9587	0.8717
SCAR(ID) w/o ref	0.9442	0.7970	0.7249	0.9696	0.8935	0.8544
SCAR(OOD)	0.9416	0.9344	0.8884	0.8887	0.9115	0.8574
	StackExchange					
SCAR(ID)	0.9020	0.8574	0.6867	-0.4330	0.9646	-0.4803
SCAR(ID) w/o rl	0.9274	0.8224	0.6968	0.7312	0.8978	0.4480
SCAR(ID) w/o ref	0.9778	0.8844	0.8660	0.9836	0.9143	0.8952
SCAR(OOD)	0.9702	0.8502	0.8249	0.7451	0.0083	-0.1289

Table 18: Cosine similarities between linguistic form representations ($\mathbf{v}_p$) and instructional surprisal representations ($\mathbf{v}_c$) for "direct", "referenced", and human-written responses. The table reports the cosine similarities between (1) "direct" and "referenced" responses, (2) "referenced" and human-written responses, and (3) "direct" and human-written responses, separately for linguistic form and instructional surprisal representations. These similarities are computed using representations from SCAR rankers trained with different configurations: SCAR(ID) trained on in-domain data, SCAR(ID) without representation learning regularization (w/o rl), SCAR(ID) without "referenced" responses (w/o ref), and SCAR(OOD) trained on out-of-domain data. The SCAR rankers are applied to response triplets generated for the same instructions in the LIMA and StackExchange datasets. Results are reported separately for each dataset, with higher cosine similarity values indicating greater alignment between the respective representations.

our optimization objectives for representation similarities. However, when SCAR training excludes "referenced" responses or utilizes out-of-domain data, these expected similarity patterns are significantly disrupted. Consequently, the performance of the META-LLAMA-3-8B model deteriorates when fine-tuned on data selected by such SCAR configurations.

In summary, incorporating "referenced" responses and utilizing in-domain data during SCAR training are crucial for maintaining the desired representation similarities. These findings emphasize the importance of carefully curating training data within SCAR to effectively model both linguistic form and instructional surprisal. This approach ensures robust SCAR data selection performance and, ultimately, enhances LLM performance across different domains.

E Bias Analysis

We categorize bias into two types–fairness bias and lexical bias (Vanmassenhove et al., 2021)–and conduct separate experiments to evaluate each.

E.1 Fairness Bias Analysis.

Model	Data Type	Data Size	Regard Diff. (Positive + Negative, % ↓)	Toxicity Ratio (Male, % ↓)	Toxicity Ratio (Female, % ↓)
Meta-LLaMA-8B	Full Human Written	10k	1.03	0.97	1.66
	Subset Human Written	1k	2.33	0.00	0.83
	Full Mixed Synthetic	10k	1.63	0.28	1.66
	Subset Mixed Synthetic	1k	0.22	1.25	2.50
OLMo-7B	Full	320k	0.82	0.28	0.28
	Subset	2.5k	0.42	0.83	1.11

Table 19: Fairness and safety metrics for models trained on full datasets and subsets. Regard difference (Positive + Negative, % ↓) reflects the absolute value of the sum of positive and negative differences, with lower values (indicated by ↓) signifying better fairness. Toxicity ratios for male and female prompts (% ↓) highlight model safety, where lower values are better.

Evaluation Settings. To evaluate fairness bias, we analyze the toxicity and sentiment polarity of model responses across different demographic and occupational groups. The evaluation consists of two

Data Type	SCAR (ID)	Random	Perplexity	Superfiltering	HFR	AlpaGasus	Diversity	Longest
Human Subset	2.33	2.42	0.97	0.88	0.87	2.36	0.80	2.17
Mixed Synthetic Subset	0.22	0.75	1.04	0.38	0.82	0.16	0.62	0.28

Table 20: Regard difference results (|Positive + Negative|) for models trained on subsets selected from Human full data and Mixed Synthetic full data using different selection methods. Lower values (↓) indicate better fairness across domains.

components:

- **Gender Bias:** Using prompts from WinoBias (Zhao et al., 2018), we generate model responses and assess toxicity levels using a pretrained hate speech detection model from Vidgen et al. (2021). Lower toxicity ratios for male and female prompts (% ↓) indicate better fairness.
- **Occupational Bias:** Using prompts from BOLD (Dhamala et al., 2021), we generate model responses and evaluate language sentiment polarity with the Regard metric (Sheng et al., 2019). This analysis includes comparisons across categories such as professions (e.g., artistic versus computer occupations), gender (e.g., actors versus actresses), political ideologies (e.g., anarchism versus capitalism), race (e.g., African Americans versus Asian Americans), and religious ideologies (e.g., atheism versus Buddhism). We report the absolute value of the sum of positive and negative regard differences (% ↓), with lower values indicating reduced bias.

We compare models fine-tuned on subsets selected by various methods with those trained on full datasets, evaluating the impact of human-written and mixed synthetic subsets on fairness bias in LLM training.

Discussion. The results (Tables 19 and 20) demonstrate that SCAR-selected subsets maintain fairness while significantly reducing dataset size. For human-written data, SCAR(ID) achieves a fairness score of 2.33, which is comparable to the full dataset score of 1.03. Additionally, SCAR(ID)-selected subsets show improvements in toxicity ratios, achieving 0.00 for male prompts and 0.83 for female prompts compared to 0.97 (male) and 1.66 (female) for the full dataset, indicating its capability to maintain fairness with smaller data.

When compared to other selection methods, SCAR(ID) achieves comparable or slightly better fairness in some cases. For mixed synthetic data, SCAR(ID)-selected subsets achieve the lowest regard difference (0.22% ↓) compared to Random (0.75%) and Perplexity (1.04%). These findings confirm that SCAR maintains fairness on par with other methods while balancing data efficiency, making it an effective strategy for fine-tuning fair LLMs.

E.2 Lexical Diversity Bias Analysis.

	Methods for Data Selection								
	Full Data	SCAR (ID)	Random	Perplexity	Superfiltering	HFR	AlpaGasus	Diversity	Longest
Instruction									
TTR	29.54	27.92	30.04	30.04	30.63	27.18	29.32	32.78	33.57
MTLD	14.71	14.72	14.77	14.83	14.80	14.61	14.85	14.71	14.69
Response									
TTR	23.37	16.60	23.22	22.37	21.79	18.09	23.13	24.69	5.35
MTLD	14.43	14.40	14.44	14.53	14.31	14.52	14.55	14.40	13.77

Table 21: Lexical diversity metrics (TTR and MTLD) for instructions and responses within different datasets, either the full open-domain human-written dataset (**Full Data**) or subsets with 2500 examples selected using various data selection methods: **SCAR (ID)**, **Random**, **Perplexity**, **Superfiltering**, **HFR**, **AlpaGasus**, **Diversity**, and **Longest**.

Evaluation Settings. We measure lexical bias in instructions and responses separately using two complementary metrics: TTR and MTLD. **Type-Token Ratio (TTR)** measures the ratio of unique words (types) to the total number of words (tokens) in a text. Higher TTR values indicate a greater immediate variety of words, making it sensitive to text length; shorter texts typically have higher TTR scores as they are less likely to repeat words. **Measure of Textual Lexical Diversity (MTLD)**, on the other hand, evaluates how lexical diversity is maintained throughout an entire text. It considers how often unique words appear relative to repeated words across longer segments, offering a more robust and length-independent view of lexical richness. We apply these metrics to the full open-domain human-written dataset (Full Data) and to 2,500-example subsets selected by various methods—SCAR (ID), Random, Perplexity, Superfiltering, HFR, AlpaGasus, Diversity, and Longest—to understand how each selection method influences lexical diversity.

Discussion. As shown in Table 21, SCAR-selected subsets exhibit slightly reduced lexical diversity in responses, indicated by lower TTR values, decreasing from 23.4 to 16.6 compared to the full dataset. We conjecture this is due to SCAR's focus on instructional surprisal consistency. As shown in Table 11, SCAR enhances the consistency of linguistic forms (lower standard deviations of TTR) in selected responses without affecting their mean TTR. This indicates that the reduced response-level TTR is likely due to instructional surprisal consistency rather than consistency in linguistic forms. In contrast, **the impact on instructions is less pronounced**, with TTR decreasing only slightly from 29.5 to 28 compared to the full dataset, indicating that SCAR does not significantly limit the coverage of instructional content.

Despite these shifts in TTR, our MTLD scores remain comparable to both the full dataset and other selection methods, for both instructions and responses. In other words, while the immediate variety of word choices (as reflected by TTR) decreases, the overall, sustained richness of vocabulary (as measured by MTLD) is preserved. Since instruction-level diversity is more crucial for LLM fine-tuning performance (Lu et al., 2023; Bukharin and Zhao, 2023), SCAR-selected subsets still preserve the kind of lexical variety that matters most. These findings align with results from the extensive experiments in the main body of the paper, where SCAR-selected subsets continue to achieve strong performance.

F Response Examples

F.1 Abbreviated Example

For the same instruction, we generated three style variations—human-written, referenced, and direct—to investigate how linguistic form and instructional surprisal influence LLM performance. Table 22 provides abbreviated examples of "human", "referenced", and "direct" responses to the instruction. The detailed content in this table has been condensed to focus on highlighting the stylistic differences. From Table 22, we can observe the shared aspects of linguistic form and instructional surprisal across the three distinct styles of rewrites.

Instruction
Why is FIFA against adding instant replay to the game?
Human Response
There are a few fundamental considerations: Uncertainty is a part of the game. ... Football is global. ... Football is a fast-paced game with few opportunities for stoppage. ... In 2008, FiFA President Sepp Blatter said: ...
Human Referenced Response from GPT-3.5-turbo
FIFA is against adding instant replay to the game for several reasons. One of the main concerns is that uncertainty is seen as a natural part of the game,... Additionally, football is a global sport;... Another factor is that football is a fast-paced game with minimal stoppages;...
Direct Response from GPT-3.5-turbo
FIFA is against adding instant replay to the game because they believe it would disrupt the flow of the game ... They also argue that human error is a part of the game ... Additionally, implementing instant replay would require significant changes to the rules and regulations ...

Table 22: Examples of different response types for a given instruction. Some details are abbreviated as '...'. Shared surprisal-related style elements between "Human" and "Referenced" responses are highlighted in orange, and shared linguistic form elements between "Referenced" and "Direct" responses are in green.

F.2 Complete Example

Table 23 provides complete examples of human, "referenced," and "direct" responses to the instruction, preserving all details. As shown, the "referenced" response closely aligns with the semantic content of the human response, thereby reflecting comparable instructional surprisal. In contrast, the "direct" response maintains similar transitional phrases and formatting to the "referenced" response, indicating a shared linguistic form between them.

Instruction
Why is FIFA against adding instant replay to the game?
Human Response
There are a few fundamental considerations: Uncertainty is a part of the game. The referee and linesmen are the arbiters of the action. Instant replay would limit their authority. Football is global. Many places do not have access to the technology necessary to implement instant replay. Football is a fast-paced game with few opportunities for stoppage. Instant replay would unnecessarily slow the pace. In 2008, FiFA President Sepp Blatter said: "Let it be as it is and let's leave [football] with errors. The television companies will have the right to say [the referee] was right or wrong, but still the referee makes the decision – a man, not a machine." Blatter and others at FIFA have repeatedly argued against using instant replay. But lately, FIFA has begun to come around on the idea. At FIFA's Congress in São Paulo prior to the 2014 World Cup, Blatter proposed allowing managers 2 challenges per game, much like the NFL handles challenges today. Although it's not quite "instant replay," in 2012 FIFA approved two different goal-line technologies to improve referee accuracy: "Hawk-Eye" – which uses visual triangulation to determine ball location – and "GoalRef" – which uses a microchip in the ball and magnetic sensors within goalposts. (Many say FIFA approved the technologies in response to Frank Lampard's snubbed "goal" against Germany in the 2010 World Cup.) In 2013, FIFA announced it would use GoalControl, a German competitor to Hawk-Eye, at the Confederations Cup, basically a rehearsal for the World Cup. After a successful run at that tournament, FIFA announced it would implement GoalControl at the 2014 World Cup. The Guardian provides a pretty helpful FAQ on the technology. FIFA may choose to implement further changes to approved goal-line technology and referee rules when it meets again after the World Cup.
Human Referenced Response from GPT-3.5
FIFA is against adding instant replay to the game for several reasons. One of the main concerns is that uncertainty is seen as a natural part of the game, and implementing instant replay would limit the authority of the referees and linesmen. Additionally, football is a global sport, and many regions may not have the necessary technology to support instant replay. Another factor is that football is a fast-paced game with minimal stoppages, so introducing instant replay could potentially disrupt the flow of the game. Ultimately, FIFA has historically been hesitant to embrace instant replay, preferring to maintain the human element in decision-making on the field.
Direct Response from GPT-3.5
FIFA is against adding instant replay to the game because they believe it would disrupt the flow of the game and potentially lead to delays. They also argue that human error is a part of the game and adding instant replay could take away from the excitement and unpredictability of live sports. Additionally, implementing instant replay would require significant changes to the rules and regulations of the game, which FIFA may be hesitant to do.

Table 23: Comparison of the three responses regarding FIFA's stance on instant replay.