SimCT: Recovering Lost Supervision for Cross-Tokenizer On-Policy Distillation

Jie Sun[1,2,3,*,‡] **Mao Zheng**[2,*] **Mingyang Song**[2,*] **Qiyong Zhong**[1]
Yilin Cheng[4] **Bichuan Feng**[4] **Pengfei Liu**[3] **Junfeng Fang**[1,†] **Xiang Wang**[1,†]

[1] University of Science and Technology of China [2] Foundation Model Department, Tencent
[3] Shanghai Innovation Institute [4] Zhongguancun Academy

sunjie2019@mail.ustc.edu.cn

Abstract

On-policy distillation (OPD) is a standard tool for transferring teacher behavior to a smaller student, but it implicitly assumes that teacher and student predictions are comparable token by token, an assumption that fails whenever the two models tokenize the same text differently. Under heterogeneous tokenizers, exact shared-token matching silently discards a large fraction of the teacher signal at precisely the positions where vocabularies disagree. We propose **Simple Cross-Tokenizer OPD (SimCT)**, which restores this signal by enlarging the supervision space: alongside shared tokens, SimCT compares teacher and student over short multi-token continuations that both tokenizers can realize, leaving the OPD loss form itself unchanged. We show that these units form the finest interface consistent with the two fixed tokenizations, and that coarser alternatives remove teacher-student distinctions that are useful for on-policy learning. Across three heterogeneous teacher-student pairs on mathematical reasoning and code-generation benchmarks, SimCT shows consistent gains over shared-vocabulary OPD and representative cross-tokenizer baselines, with ablations supporting the benefit of recovering supervision discarded by exact shared-token matching. Code is available at https://github.com/sunjie279/SimCT-.

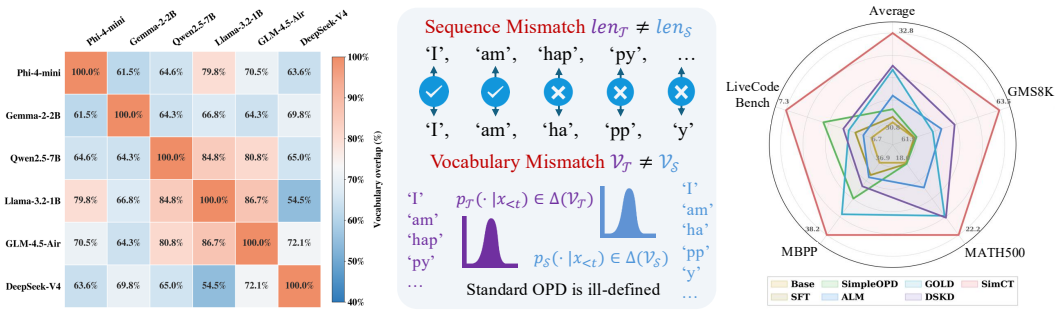
1 Introduction

Knowledge distillation transfers the behavior of a strong teacher to a smaller student and remains central to efficient LLM deployment [1, 2, 3, 4]. For autoregressive LLMs, however, conventional offline knowledge distillation trains the student on fixed contexts rather than on states induced by its own generations [5, 6], so early generation errors can compound and move the student into contexts that were rarely supervised during training [7, 8, 5, 9]. On-policy distillation (OPD) addresses this train–test mismatch by querying the teacher on prefixes sampled from the student's policy, following the broader principle of on-policy imitation learning [10, 11, 12, 13, 14]. This makes OPD a natural paradigm for LLM distillation [15, 16, 17, 18, 19], but it also exposes a hidden requirement for cross-model supervision: at each student-generated prefix, teacher and student predictions must be defined over comparable prediction units.

This requirement is non-trivial in realistic LLM distillation, where teacher and student models come from different families and use different tokenizers [20]. The mismatch arises at two levels. First, at the vocabulary level: as shown in Figure 1(A), modern LLM tokenizers share only a limited portion of their vocabularies across model families, so restricting supervision to tokens that appear in both vocabularies excludes teacher predictions outside this intersection. Second, at the sequence level: the same text may be segmented into different token sequences, so the teacher's next-token

[*] Equal Contribution. [†] Corresponding Authors. [‡] Work done during internship at Tencent.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(A) Limited Vocabulary Overlap (B) Tokenizer Mismatch Breaks OPD (C) SimCT Recovers Lost Supervision

Figure 1: **Motivation and performance of SimCT. (A)** LLM tokenizers often share only partial vocabularies across model families, making shared-token distillation restrictive. **(B)** The same text can induce different token boundaries and prediction spaces, so standard token-level OPD is ill-defined. **(C)** SimCT constructs a common aligned supervision space and achieves the best average Pass@1 over SFT and prior cross-tokenizer baselines. Each axis is independently normalized for visualization; absolute results with standard deviations are reported in Table 1.

distribution at a student-generated prefix may not supervise the same text unit as the student's next-token distribution [21, 22, 23, 24, 25]. As illustrated in Figure 1(B), this sequence mismatch breaks a basic assumption of standard OPD, namely that teacher and student predict over the same text unit at every supervised position.

Existing cross-tokenizer distillation methods use several kinds of supervision interfaces. Distribution-alignment methods compare heterogeneous logits through optimal transport [21, 26], while GOLD adapts distribution alignment to on-policy training through probability merging and a hybrid vocabulary loss [27]. Representation-transfer methods use learned cross-model projections [22, 28], introducing additional trainable parameters. Other interfaces use chunk likelihoods [29], BPE structure [25], dynamic vocabulary mapping [30], or byte-level decoders [24]. These approaches differ in granularity, cost, and training setting. Our focus is a parameter-free interface that recovers supervision on student-generated prefixes using the existing tokenizer boundaries. This raises a natural question: how simple can the supervision interface be while still recovering most of the teacher signal that tokenizer mismatch would otherwise discard?

We propose **Simple Cross-Tokenizer OPD (SimCT)**, a cross-tokenizer OPD method that changes the supervision space rather than the OPD objective. Instead of forcing teacher and student logits to match only on identical tokens, SimCT induces a normalized preference distribution over a common supervision space and applies the standard reverse-KL OPD loss there [12, 31]. The space contains shared tokens and *minimal aligned units* recovered from tokenizer-mismatched text. A minimal aligned unit is the shortest non-empty span bounded by synchronized boundaries of the two fixed tokenizations, even when their token sequences differ. SimCT requires no additional parameters, no learned projections, and no transport computation, since the supervision space is constructed directly from the two existing tokenizers. It recovers teacher signal that shared-vocabulary OPD discards. SimCT and SimpleOPD share the reverse-KL loss form; SimCT additionally uses a normalization safeguard specified in Appendix C.2. The ablations examine the benefit of the recovered supervision within this training design. As previewed in Figure 1(C), this supervision-space expansion substantially improves over existing cross-tokenizer OPD baselines.

We evaluate SimCT across three model families on mathematical reasoning and code-generation benchmarks. SimCT consistently improves over SFT and representative cross-tokenizer OPD baselines, achieving an average relative gain of **6.0%** over SFT with modest additional computational cost relative to shared-vocabulary OPD. Ablations further explain where the gains come from: increasing recovered tokenizer-mismatched text supervision improves performance, whereas coarsening minimal aligned units removes KL supervision signal and reduces downstream gains. These trends support our theoretical analysis, which characterizes SimCT as the finest boundary-consistent supervision interface jointly expressible by the teacher and student tokenizers, and shows that coarser interfaces can erase useful within-text teacher–student distinctions. Overall, our results identify supervision-space construction as a key design choice in cross-tokenizer OPD, with minimal aligned units offering a simple way to recover teacher supervision otherwise lost under exact token overlap.

2 Related Work

Knowledge distillation and on-policy distillation. Knowledge distillation transfers teacher behavior to a smaller student and has become a standard tool for efficient LLM deployment [1, 3], with early work treating it as offline compression on fixed corpora [32, 33, 6]. Because fixed training contexts can differ from the prefixes the student visits at inference [7, 5], on-policy distillation (OPD) instead queries the teacher on student-generated prefixes [34, 11], while related distillation methods refine the objective and data use [12, 13, 14, 15, 35, 36, 37]. A complementary line analyzes *when* and *where* OPD provides signal: distributional compatibility governs success or failure at the sequence level [18], while token-level studies show that supervision quality varies sharply across positions [38, 39]. Recent OPD methods also refine distinct components of the supervision pipeline. TrustMOPD allocates token-level supervision among specialist teachers using label-free reliability estimates [40]; influence-directed distillation controls the effect of updates on sampled-token probabilities [41], and SOD adapts supervision at the reasoning-step level [42]. Guided-OPD changes the rollout distribution through curriculum-scheduled teacher guidance in multi-turn interactions [43], whereas Data-Free OPD generates its own training questions to reduce dependence on external prompt corpora [44]. EasyOPD provides modular infrastructure for studying supervision forms, tokenizer compatibility, and optimization granularity [45]. These directions address teacher selection, signal weighting, rollout construction, data sourcing, and implementation. SimCT instead addresses the comparison units needed when tokenizers differ. These complementary design choices suggest extensions of cross-tokenizer OPD.

Cross-tokenizer knowledge distillation. When teacher and student tokenize the same text differently, prior work bridges the vocabulary gap at one of several interfaces. Optimal-transport methods align heterogeneous logits directly, either between token distributions [21], jointly at token and sequence levels [26], or layer-wise with chain-of-thought augmentation [46]. Dual-space and representation-matching methods instead unify the output spaces themselves through cross-model attention and its variants [22, 47]. Other methods construct comparison interfaces through approximate chunk likelihoods [29], doing context-aware sequence matching with dynamic vocabulary mapping [30], or using bytes as a tokenizer-agnostic interface [24], while related directions reuse the teacher loss as supervision [48], extend cross-tokenizer transfer to preference alignment [23], or bridge tokenizer gaps in embedding-model representation space [49].

3 Methodology

3.1 Problem Setup

Let $\mathcal{T}$ and $\mathcal{S}$ denote the teacher and student models, with vocabularies $\mathcal{V}_T$ and $\mathcal{V}_S$. In on-policy distillation, supervision is provided on prefixes sampled from the student policy rather than a fixed offline dataset. At step t, let $x_{<t}$ denote the student-generated prefix. Both models are conditioned on the same text-level context $x_{<t}$, but they predict the next token over their own vocabularies:

$$p_T(\cdot \mid x_{<t}) \in \Delta(\mathcal{V}_T), \qquad p_S(\cdot \mid x_{<t}) \in \Delta(\mathcal{V}_S). \tag{1}$$

Here, $p_T(v \mid x_{<t})$ is the probability the teacher assigns to token $v \in \mathcal{V}_T$, and $p_S(u \mid x_{<t})$ is defined analogously for student token $u \in \mathcal{V}_S$. We write $\Delta(\mathcal{V})$ for the probability simplex over $\mathcal{V}$. When teacher and student share a tokenizer, the two distributions in Eq. (1) are defined over a common prediction space, and standard OPD compares them directly, e.g., by minimizing a token-level KL:

$$\mathcal{L}_{\mathrm{OPD}}(x_{<t}) = \mathrm{KL}(p_S(\cdot \mid x_{<t}) \,\|\, p_T(\cdot \mid x_{<t}))\,. \tag{2}$$

This objective is well defined only because each student token corresponds to the same prediction unit as the corresponding teacher token. Thus, standard OPD implicitly assumes that teacher and student probabilities live on the same simplex and can be compared token by token.

Cross-tokenizer OPD breaks this assumption in two coupled ways. First, the vocabularies differ, $\mathcal{V}_T \neq \mathcal{V}_S$, so the teacher and student distributions in Eq. (1) are supported on different probability spaces. Second, the same text may be segmented into different token sequences, so textually equivalent texts do not necessarily admit a one-to-one token alignment. We refer to these obstacles as *vocabulary mismatch* and *sequence mismatch*, respectively. The goal of cross-tokenizer OPD is to construct a common supervision interface in which teacher feedback remains comparable to the student's next-token prediction on student-generated prefixes.

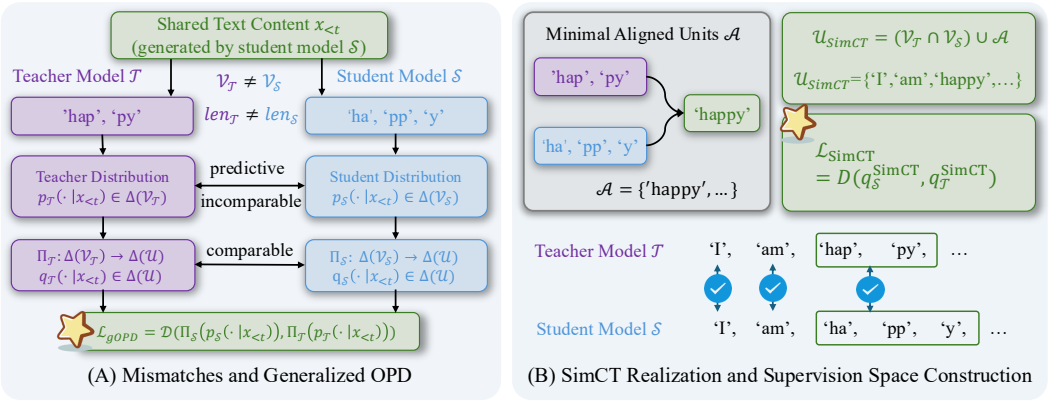


Figure 2: **Framework of SimCT. (A)** Teacher and student distributions are mapped to a common supervision space $\mathcal{U}$ under the same student-generated prefix. The panel shows the next-token form of the interface; for multi-token candidates, Eqs. (3)–(4) define scoring through the autoregressive conditional families. **(B)** SimCT builds $\mathcal{U}_{\text{SimCT}} = (\mathcal{V}_T \cap \mathcal{V}_S) \cup \mathcal{A}$ by adding minimal aligned units $\mathcal{A}$, then applies the original OPD loss on the induced supervision distributions.

3.2 Generalized OPD over a Common Supervision Space

Cross-tokenizer OPD requires comparing two next-token distributions that are defined over different vocabularies and may correspond to different text segmentations. We handle this by replacing the shared-vocabulary assumption with a shared set of prediction units. Figure 2(A) sketches the next-token form of this interface; the formulation below extends scoring to multi-token candidates. Specifically, we introduce a *common supervision space* $\mathcal{U}$ and scoring maps. Let $\mathcal{P}_M(h) = \{p_M(\cdot \mid h, z) : z \in \mathcal{V}_M^*\}$ denote model M's autoregressive conditional family after prefix h, including the empty continuation z. The maps act on this family:

$$\Pi_T : \mathcal{P}_T(h) \mapsto q_T(\cdot \mid h) \in \Delta(\mathcal{U}), \qquad \Pi_S : \mathcal{P}_S(h) \mapsto q_S(\cdot \mid h) \in \Delta(\mathcal{U}), \tag{3}$$

with the candidate set held fixed when scoring. For multi-token units, access to conditional probabilities inside each candidate is required; a single next-token probability vector is insufficient. Given a student-generated prefix $x_{<t}$, these maps induce comparable distributions

$$q_T(\cdot \mid x_{<t}) = \Pi_T(\mathcal{P}_T(x_{<t})), \qquad q_S(\cdot \mid x_{<t}) = \Pi_S(\mathcal{P}_S(x_{<t})). \tag{4}$$

Distillation can then be performed directly in $\mathcal{U}$:

$$\mathcal{L}_{\text{gOPD}}(x_{<t}) = \mathcal{D}(q_S(\cdot \mid x_{<t}), q_T(\cdot \mid x_{<t})), \tag{5}$$

where $\mathcal{D}$ denotes a distillation divergence over the common supervision space.

This formulation recovers standard OPD as a special case. When teacher and student share a tokenizer, we can set $\mathcal{U} = \mathcal{V}_T = \mathcal{V}_S$ and $\Pi_M(\mathcal{P}_M(h)) = p_M(\cdot \mid h)$, reducing Eq. (5) to the token-level objective in Eq. (2). In the cross-tokenizer setting, however, the central design choice is how to construct $\mathcal{U}$ so that it preserves useful teacher supervision while remaining compatible with student predictions. A direct baseline is to use only the shared vocabulary $\mathcal{U}_{\text{shared}} = \mathcal{V}_T \cap \mathcal{V}_S$, and restrict both distributions to this subspace. We refer to this baseline as **SimpleOPD**. SimpleOPD is a natural but conservative instantiation of Eq. (5): it avoids comparing tokens that are not shared by both vocabularies, but it discards supervision signal from texts that are semantically comparable yet represented by different token segments. Thus, SimpleOPD fails not because generalized OPD is inadequate, but because exact token overlap defines an overly narrow supervision space.

3.3 SimCT: Recovering Supervision via Minimal Aligned Units

To expand the common supervision space that goes beyond exact token overlap, we propose **Simple Cross-Tokenizer OPD (SimCT)**. Its central principle is to recover teacher supervision whenever teacher and student predictions refer to the same underlying text, even if the two tokenizers realize that text with different token boundaries. Figure 2(B) gives a simple example: the teacher tokenizes

"happy" as "hap", "py", while the student tokenizes the same text as "ha", "pp", "y". Although not comparable as individual tokens, these refer to the same text, so SimCT treats it as a recoverable supervision unit. To construct these units, SimCT considers texts where the two tokenizations disagree and forms the finest boundary-consistent text units realizable by both tokenizers. We call these units *minimal aligned units* (MAU), and denote the set of all such units by $\mathcal{A}$. SimCT then defines its supervision space as

$$\mathcal{U}_{\mathrm{SimCT}} = (\mathcal{V}_T \cap \mathcal{V}_S) \cup \mathcal{A}. \tag{6}$$

The shared-vocabulary term keeps supervision that is already comparable on 1:1-aligned shared tokens, while $\mathcal{A}$ recovers supervision from texts that tokenize differently but are comparable after mapping to common aligned units.

We now define the SimCT scoring maps Π_M^{SimCT} for $M \in \{\mathcal{T}, \mathcal{S}\}$. SimCT first assigns each unit $u \in \mathcal{U}_{\mathrm{SimCT}}$ a continuation score and then normalizes over the candidate supervision set. For a 1:1-aligned shared token $u \in \mathcal{V}_T \cap \mathcal{V}_S$, we use the original next-token log-probability, $s_M(u \mid x_{<t}) = \log p_M(u \mid x_{<t})$. For a minimal aligned unit $u \in \mathcal{A}$, let $\tau_M(u) = (v_1, \ldots, v_k)$ be its tokenizer-M realization, and define

$$s_M(u \mid x_{<t}) = \frac{1}{k} \log p_M(u \mid x_{<t}) = \frac{1}{k} \sum_{j=1}^{k} \log p_M(v_j \mid x_{<t}, v_{<j}). \tag{7}$$

The factor $1/k$ uses the average log-likelihood rather than the raw sequence log-likelihood, avoiding systematically penalizing units that require more tokens under a given tokenizer [50]. This normalization is a standard practical choice, not a requirement of the theoretical construction in §3.4. The normalized supervision distribution over the candidate set is then

$$q_M^{\mathrm{SimCT}}(u \mid x_{<t}) = \frac{\exp(s_M(u \mid x_{<t}))}{\sum_{u' \in \mathcal{U}_{\mathrm{SimCT}}} \exp(s_M(u' \mid x_{<t}))}, \tag{8}$$

which we denote as $q_M^{\mathrm{SimCT}} = \Pi_M^{\mathrm{SimCT}}(\mathcal{P}_M)$. Equation (8) defines a distribution over the finite candidate supervision set. Its probabilities are obtained by normalizing continuation scores: shared tokens retain their next-token log-probabilities, while multi-token MAUs use model-specific length-normalized continuation likelihoods. This candidate-score normalization and the original model's next-token probability measure operate on distinct spaces. Among retained shared tokens, the common normalizer preserves probability ratios. Multi-token MAU scores depend on the conditional factors and token counts of each model's realization; Appendix H.1 details these scoring properties.

The SimCT loss is the generalized OPD loss instantiated on this recovered supervision space:

$$\mathcal{L}_{\mathrm{SimCT}}(x_{<t}) = \mathcal{D}\big(q_S^{\mathrm{SimCT}}(\cdot \mid x_{<t}), q_T^{\mathrm{SimCT}}(\cdot \mid x_{<t})\big)\,. \tag{9}$$

SimCT performs OPD over the next continuation units rather than tokenizer-specific next tokens. For a multi-token unit, Eq. (7) scores the candidate continuation under the current prefix $x_{<t}$ through its autoregressive factors $p_M(v_j \mid x_{<t}, v_{<j})$, using only the prefix and the candidate's preceding tokens as conditioning. Candidate construction can use the already sampled student rollout, as in Algorithm 1; it does not require an external ground-truth continuation. Thus, SimCT preserves the on-policy training form while changing the supervision interface from exact token overlap to aligned continuation units. For all main SimCT experiments, we mask positions where the reciprocal of the teacher score normalizer in Eq. (8) exceeds 2; implementation details are in Appendix C.2. The procedure is summarized in Algorithm 1.

3.4 Properties of the SimCT Supervision Space

We present two theoretical results that clarify the design of the SimCT supervision space. First, minimal aligned units provide the finest comparable text units consistent with the two fixed tokenizations. Second, merging these units into coarser supervision units can remove teacher–student distinctions that are visible at the minimal-unit level. Formal definitions, assumptions, and proofs are provided in Appendix E.

Theorem 1 (Minimal aligned units)**.** *For a text region with fixed teacher and student tokenizations, SimCT constructs a partition into minimal aligned units. Each unit is a union of consecutive complete tokens under both fixed tokenizations. No unit admits a strictly finer non-empty decomposition with this property.*

Theorem 1 shows that SimCT constructs the finest boundary-consistent interface on which teacher and student predictions can be compared, making the supervision space more expressive than exact shared-token matching while avoiding unconstrained character-level comparison.

Proposition 1 (Signal loss under coarser supervision spaces)**.** *For a fixed student-generated prefix, let $q_T^{\min}$ and $q_S^{\min}$ be teacher and student distributions over SimCT minimal aligned units, with $q_T^{\min}(u) > 0$ whenever $q_S^{\min}(u) > 0$. Let $q_T^{\mathcal{C}}$ and $q_S^{\mathcal{C}}$ be the distributions obtained by a non-trivial coarsening $\mathcal{C}$, which sums probabilities inside each coarse unit. Then*

$$\mathrm{KL}\big(q_S^{\min} \,\|\, q_T^{\min}\big) \geq \mathrm{KL}\big(q_S^{\mathcal{C}} \,\|\, q_T^{\mathcal{C}}\big)\,. \tag{10}$$

The gap is exactly the within-unit teacher–student discrepancy removed by the coarsening.

Proposition 1 motivates SimCT to preserve minimal aligned units rather than coarser units. In its idealized distributional setting, coarsening removes the within-unit teacher–student discrepancy that provides a fine-grained distillation signal; Section 4.4 measures this erased signal by the KL gap over student-generated prefixes. Together with Theorem 1, this supports using the finest tokenizer-compatible supervision units. The practical SimCT objective in Eq. (8) implements this principle with normalized continuation scores over a finite candidate set. The KL result applies to distributions coarsened by summing probabilities within fixed groups. Concatenating strings and recomputing continuation scores defines a separate transformation.

4 Experiments

4.1 Experimental Setup

Models and pairs. We construct three teacher–student pairs that span all three model families and tokenizers: Qwen2.5-7B-Instruct [51] → Phi-4-mini-Instruct [52], Qwen → Gemma-2-2B-IT [53], and Phi → Gemma. All pairs follow the realistic large-to-small distillation direction. Two distinct models (Qwen and Phi) serve as teachers and two distinct models (Phi and Gemma) serve as students across pairs, so our results do not hinge on any single teacher or student choice.

Data and warm-start. We construct teacher-specific corpora to warm-start the students before OPD. Prompts are drawn from GSM8K [54], Orca-Math [55], OpenMathInstruct-1 [56], and MATH excluding MATH-500 [57] for math, and OpenCodeInstruct [58], KodCode [59], TACO [60], and CodeContests [61] for code. For each prompt, the teacher samples 8 responses (temperature 0.6, top-$p = 0.95$); we keep one per prompt via task-specific correctness checks where available and quality filters otherwise, yielding a 10K-example SFT corpus for each teacher. Within each teacher–student pair and training seed, all OPD methods start from the same corresponding SFT checkpoint. Full composition, filtering, and license details are in Appendix B.

On-policy distillation. Starting from the warm-start checkpoint, all methods follow a common OPD loop with method-specific supervision and loss safeguards: at each step the student samples a prefix under its current policy, the teacher is queried on this prefix to produce next-token supervision, and the student is updated under each method's loss. Training prompts, prefix-generation settings, optimizer, schedule, training epochs, and hardware are held identical across methods. Hyperparameters are in Appendix C.

Baselines. We compare SimCT against four baselines that span the main families of cross-tokenizer distillation. SimpleOPD performs exact shared-vocabulary OPD, supervising only the tokens that appear in both vocabularies. DSKD [22] unifies teacher and student output spaces through a learned cross-model projection, requiring additional trainable parameters. ALM [29] matches approximate likelihoods over tokenizer-induced chunks, providing supervision at the chunk level rather than the token level. GOLD [27] extends universal logit distillation [21] to on-policy training with text-aligned probability merging across tokenizer boundaries.

Benchmarks. We evaluate mathematical reasoning on GSM8K [54] and MATH-500 [62], and code generation on MBPP [63] and LiveCodeBench-v6 [64]. For each SFT or OPD configuration, we train independently with 5 seeds and evaluate each resulting checkpoint once under zero-shot prompting, reporting mean pass@1 and standard deviation across the 5 runs; full templates, decoding settings, and scoring details are in Appendix D.

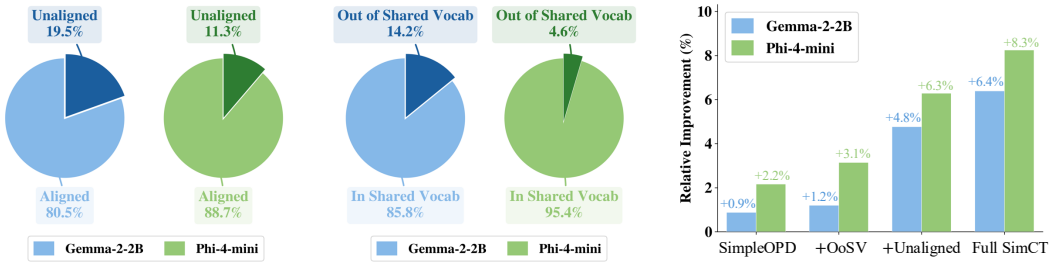
Table 1: Main results in the cross-tokenizer OPD setting. All distillation methods start from the same SFT-warm-started student checkpoint within each teacher–student pair and seed, and follow the same common training protocol. For SFT and OPD, we report mean pass@1 and standard deviation over 5 independent training runs, evaluating each checkpoint once. Base models use 5 decoding seeds. The Average column is the arithmetic mean of the four displayed benchmark means, rounded to two decimal places. Best and second-best student results within each teacher–student pair are shown in **bold** and underlined.

Model	Method	Math		Code		Average
		GSM8K	MATH500	MBPP	LCB-v6	
Gemma-2-2B-it	Base	61.85 ±0.25	17.96 ±0.38	36.92 ±0.30	6.65 ±0.23	30.85
Phi-4-mini-instruct	Base	85.75 ±0.31	54.56 ±0.49	55.36 ±0.39	7.92 ±0.30	50.90
Qwen2.5-7B-Instruct	Base	91.43 ±0.33	66.20 ±0.54	65.80 ±0.43	13.71 ±0.37	59.29
Qwen2.5-7B-Instruct teacher ↓ Gemma-2-2B-it student	SFT	61.86 ±0.22	18.00 ±0.35	37.14 ±0.28	6.77 ±0.20	30.94
	SimpleOPD	61.90 ±0.25	18.00 ±0.40	37.56 ±0.31	7.01 ±0.22	31.12
	ALM	62.39 ±0.27	19.40 ±0.44	37.18 ±0.30	6.71 ±0.21	31.42
	GOLD	62.22 ±0.30	21.04 ±0.48	37.84 ±0.34	6.82 ±0.23	31.98
	DSKD	62.65 ±0.31	21.16 ±0.49	37.34 ±0.32	6.86 ±0.24	32.00
	SimCT	**63.53** ±0.42	**22.16** ±0.52	**38.21** ±0.35	**7.29** ±0.25	**32.80**
Qwen2.5-7B-Instruct teacher ↓ Phi-4-mini-instruct student	SFT	86.06 ±0.28	55.16 ±0.45	55.40 ±0.36	8.57 ±0.28	51.30
	SimpleOPD	87.04 ±0.30	56.80 ±0.50	55.02 ±0.38	9.23 ±0.31	52.02
	ALM	87.81 ±0.33	59.84 ±0.56	55.12 ±0.40	10.86 ±0.37	53.41
	GOLD	88.32 ±0.34	60.12 ±0.59	55.08 ±0.41	12.06 ±0.42	53.90
	DSKD	88.63 ±0.34	59.68 ±0.55	55.78 ±0.40	11.43 ±0.39	53.88
	SimCT	**90.22** ±0.40	**61.40** ±0.60	**56.04** ±0.41	**12.57** ±0.44	**55.06**
Phi-4-mini-instruct teacher ↓ Gemma-2-2B-it student	SFT	62.32 ±0.22	18.20 ±0.36	37.48 ±0.29	6.43 ±0.20	31.11
	SimpleOPD	62.62 ±0.25	19.00 ±0.42	37.89 ±0.32	6.43 ±0.22	31.49
	ALM	62.24 ±0.25	19.80 ±0.45	37.80 ±0.33	6.63 ±0.23	31.62
	GOLD	62.52 ±0.26	20.36 ±0.47	38.82 ±0.35	6.57 ±0.24	32.07
	DSKD	62.37 ±0.26	19.80 ±0.44	38.64 ±0.34	6.57 ±0.23	31.85
	SimCT	**62.92** ±0.31	**21.08** ±0.49	**39.16** ±0.36	**7.14** ±0.27	**32.58**

4.2 Overall Effectiveness

Table 1 reports the main results under the cross-tokenizer OPD setting. Within each pair and training seed, all distillation methods start from the same SFT-warm-started student checkpoint and follow the same common training protocol. As reported in Appendix F, SimCT only slightly increases the wall-clock cost over SimpleOPD and is faster than the evaluated ALM, GOLD, and DSKD implementations. The headline 6.0% relative gain is the unweighted mean of the three pair-level improvements over SFT: 6.0%, 7.3%, and 4.7% after rounding. Each pair-level score first averages the four benchmark pass@1 values, so this summary gives equal weight to benchmarks and to model pairs. From these results, we draw the following observations:

- **Obs 1: SimCT improves cross-tokenizer OPD consistently across teachers, students, and task domains.** SimCT achieves the best average performance for all three teacher–student pairs and obtains the strongest result on every evaluated benchmark. The gains hold when Qwen2.5-7B-Instruct distills to two different student families, and also when Phi-4-mini-instruct serves as the teacher for Gemma-2-2B-it. They also appear on both mathematical reasoning and code-generation benchmarks. This consistency suggests that SimCT is not tied to a particular teacher, student, or task type, but provides a robust supervision interface for cross-tokenizer OPD.
- **Obs 2: Exact shared-token supervision is too conservative under tokenizer mismatch.** SimpleOPD uses the most direct common supervision space: the exact shared vocabulary. However, it improves only marginally over the SFT warm start across the three distillation settings. This indicates that shared tokens preserve only a limited portion of useful teacher feedback when teacher and student tokenizers differ. By adding minimal aligned units, SimCT recovers supervision from text continuations that are comparable but not represented as identical tokens, leading to consistently stronger performance than SimpleOPD.



(A) Token Alignment Ratio (B) Top-100 Shared Vocab Coverage (C) Supervision Recovery Decomposition

Figure 3: **Supervision recovery under tokenizer mismatch. (A)** Many teacher and student tokens are not aligned one-to-one, revealing supervision loss from sequence mismatch. **(B)** Even at aligned positions, high-probability teacher predictions may fall outside the shared vocabulary. **(C)** Recovering either missing source improves over Base, while Full SimCT performs best by recovering both in a common aligned-unit space. **+Out of Shared Vocab (+OoSV)** adds non-shared vocabulary supervision at aligned positions; **+Unaligned** adds supervision from non-1:1 aligned units. Panel (C) reports independent ablation runs, with relative improvement over the corresponding unadapted Base student.

- **Obs 3: Prior cross-tokenizer interfaces are useful but less reliable across settings.** ALM, GOLD, and DSKD often improve over SimpleOPD, confirming that cross-tokenizer supervision beyond exact token overlap is beneficial. However, their relative rankings vary across teacher–student pairs and benchmarks: no existing baseline gives a uniformly strongest interface. This suggests that making heterogeneous predictions comparable is necessary but not sufficient for cross-tokenizer OPD. The supervision interface must also preserve local teacher–student distinctions needed for on-policy next-token feedback, which SimCT addresses through minimal aligned units.

Overall, these observations support the central design choice of SimCT: keep the OPD training paradigm unchanged, but improve the supervision space in which teacher feedback is expressed. Minimal aligned units provide a more robust supervision interface than exact shared tokens or prior cross-tokenizer mappings, leading to consistent gains in realistic cross-tokenizer OPD settings. Appendix G presents a case study with math and code examples illustrating local reasoning or implementation errors in the generated responses.

4.3 Supervision Recovery under Tokenizer Mismatch

We next ask whether SimCT improves cross-tokenizer OPD by recovering teacher supervision that is discarded under tokenizer mismatch. Figure 3 decomposes the missing supervision, while Figure 4 tests whether recovering this supervision leads to downstream gains. The downstream results in these figures come from independent ablation runs. Their relative improvements use the corresponding unadapted Base student as the reference; Table 1 reports the main training runs, with gains measured relative to SFT. From these analyses, we draw the following observations:

- **Obs 4: Tokenizer mismatch removes useful teacher feedback through both sequence mismatch and vocabulary mismatch.** Figure 3(A) shows that a non-trivial fraction of teacher- and student-side tokens cannot be aligned one-to-one. This indicates that standard token-level matching fails to use teacher feedback on units that correspond to the same underlying text but are realized with different tokenizer boundaries. Figure 3(B) reveals a complementary failure mode: even at token-to-token-aligned positions, many high-probability teacher predictions fall outside the shared vocabulary. Together, these results show that SimpleOPD is conservative in two distinct ways: it discards supervision from non-1:1 aligned texts, and it also removes non-shared vocabulary predictions even when the current positions are aligned.
- **Obs 5: The discarded signals are beneficial rather than incidental tokenizer artifacts.** Figure 3(C) directly tests whether the missing signals identified above improve learning. Starting from SimpleOPD, **+Out of Shared Vocab** adds non-shared vocabulary supervision at token-to-token-aligned positions, while **+Unaligned** adds supervision from tokenizer-mismatched units that cannot be aligned one-to-one. Both variants improve over SimpleOPD for both student models,

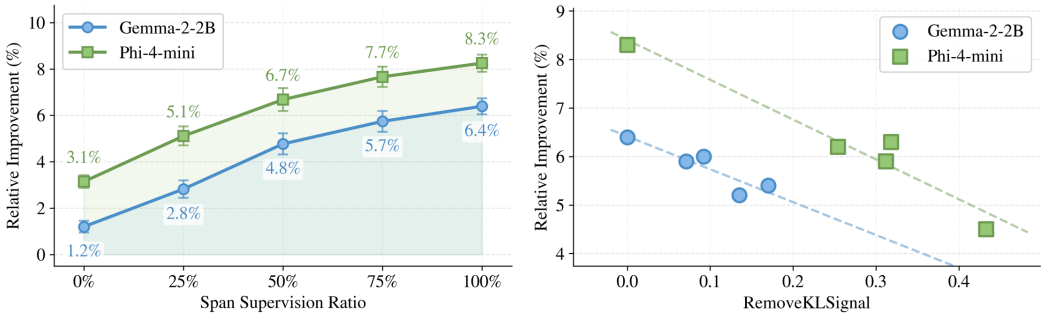


Figure 4: **Ablation on recovered unit supervision and aligned-unit coarsening. (Left)** Recovering more mismatch-unit supervision consistently improves both students. **(Right)** Larger removed within-span KL signal is generally associated with smaller gains. These are independent ablation runs; improvement is relative to the corresponding Base student. Error bars show standard deviations over 5 independent training runs, with one evaluation per checkpoint. The 0% MAU point on the left retains OoSV supervision and is therefore SimpleOPD + OoSV.

showing that the discarded supervision signal contains useful distillation information. Full SimCT achieves the largest improvement because it recovers both sources through a unified aligned-unit supervision space.

- **Obs 6: Recovering more mismatch unit supervision consistently improves downstream performance.** Figure 4 (Left) further varies the fraction of tokenizer-mismatched unit supervision incorporated into training. For both Gemma-2-2B and Phi-4-mini, performance improves as more recovered supervision is used. This trend strengthens the mechanism suggested by Figure 3: mismatch unit supervision is not merely recoverable, but provides a stable learning signal when mapped into a common supervision space. SimCT exploits this signal by aligning tokenizer-specific continuations into minimal aligned units, rather than discarding them under exact token overlap.

These findings explain why SimpleOPD yields only limited gains in Table 1. Shared-vocabulary distillation preserves only the subset of teacher feedback that is already comparable in the original token spaces. SimCT expands the supervision space to reincorporate supervision signal excluded by shared-vocabulary matching, thereby recovering useful teacher supervision while keeping the original OPD objective unchanged. Across 263,817 rollout positions, MAUs increase mean represented teacher next-token support from 95.6% to 99.5% (Appendix H.1). Additional overlap controls, same-tokenizer references, and position-level analyses are reported in Appendices H.3 and H.5.

4.4 Ablation on Supervision-Space Construction

The previous analysis shows that tokenizer-mismatched units contain useful teacher supervision. We now ask whether the *granularity* of the recovered supervision also matters. Starting from the SimCT supervision space, we construct coarsened variants by merging adjacent minimal aligned units into larger units. Let $q_T^{\min}$ and $q_S^{\min}$ denote the teacher and student distributions over minimal aligned units. For a coarsening $\mathcal{C}$, the corresponding coarse distributions aggregate the probability mass of all minimal units inside each coarse unit:

$$q_T^{\mathcal{C}}(c \mid h) = \sum_{u \in c} q_T^{\min}(u \mid h), \qquad q_S^{\mathcal{C}}(c \mid h) = \sum_{u \in c} q_S^{\min}(u \mid h), \tag{11}$$

where h is the student-generated prefix. This aggregation preserves the total probability assigned to each coarse unit, but collapses the relative teacher–student preferences among its constituent minimal units. We quantify the erased supervision signal as:

$$\Delta_{\mathcal{C}} = \mathbb{E}_h \left[\mathrm{KL}\big(q_S^{\min}(\cdot \mid h) \,\|\, q_T^{\min}(\cdot \mid h)\big) - \mathrm{KL}\big(q_S^{\mathcal{C}}(\cdot \mid h) \,\|\, q_T^{\mathcal{C}}(\cdot \mid h)\big) \right]. \tag{12}$$

Thus, $\Delta_{\mathcal{C}}$ measures the within-unit KL signal removed when minimal aligned units are merged.

- **Obs 7: Coarsening removes useful local supervision.** Figure 4 (Right) shows a negative relationship between removed KL signal and downstream improvement: coarsenings with larger $\Delta_{\mathcal{C}}$

tend to yield weaker gains for both Gemma-2-2B-it and Phi-4-mini. This indicates that coarsening does more than simplify the supervision space; it erases local teacher–student preferences useful for OPD. The result supports Proposition 1 and explains why SimCT keeps minimal aligned units instead of merging them into coarser units.

Overall, this ablation shows that SimCT benefits not only from recovering supervision under tokenizer mismatch, but also from preserving it at the finest granularity jointly expressible by both tokenizers. An additional within-unit scoring ablation favors token-length normalization over unnormalized and byte-normalized scores on Qwen2.5 $\rightarrow$ Gemma-2 (Appendix H.4).

5 Limitations

Our main evaluation centers on math and code reasoning with three teacher–student pairs spanning two teacher families and heterogeneous tokenizers; extending the study to additional domains, scales, and tokenizer families is left to future work. Methodologically, SimCT relies on mapping tokenizer mismatches to minimal aligned units, which is natural for standard BPE-style tokenizers and may require adaptation for tokenizers with substantially different segmentation behavior. Finally, we target the white-box OPD setting in which teacher next-token distributions are available; extending supervision-space construction to black-box teachers that expose only samples or scalar feedback is a promising direction.

6 Conclusion

We studied cross-tokenizer OPD, where teacher and student policies follow OPD but use different tokenizers. We argued that cross-tokenizer OPD should be viewed as a supervision-space construction problem: useful teacher feedback is lost when heterogeneous token-level predictions are forced to match only on exact shared tokens. We introduced SimpleOPD as a conservative shared-vocabulary baseline and proposed SimCT, which expands the common supervision space with minimal aligned units while preserving the standard OPD objective. Our theoretical analysis characterizes these units as the finest supervision interface consistent with the two fixed tokenizations and shows that coarsening them can erase useful teacher–student distinctions. Across model families and math/code benchmarks, SimCT consistently improves over SimpleOPD and representative cross-tokenizer distillation baselines. These results suggest that changing the supervision space, rather than the OPD training paradigm, is an effective path for cross-tokenizer OPD.

Acknowledgments and Disclosure of Funding

This research is supported by the National Natural Science Foundation of China (U25A20445, 62625608), and the National Research Foundation, Singapore under its National Large Language Models Funding Initiative (AISG Award No: AISG-NMLP-2024-002). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv preprint*, arXiv:1503.02531, 2015.

[2] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics (TACL)*, 2024.

[3] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint*, arXiv:2402.13116, 2024.

[4] Chuanpeng Yang, Wang Lu, Yao Zhu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *arXiv preprint*, arXiv:2407.01885, 2024.

[5] Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 6121–6133, Online Event, 2020.

[6] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

[7] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*, 2015.

[8] Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 4334–4343, Florence, Italy, 2019.

[9] Chris Cundy and Stefano Ermon. SequenceMatch: Imitation learning for autoregressive sequence modelling with backtracking. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*, Vienna, Austria, 2024.

[10] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pages 627–635, Fort Lauderdale, FL, USA, 2011.

[11] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*, Vienna, Austria, 2024.

[12] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*, Vienna, Austria, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

[13] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, volume 235, Vienna, Austria, 2024.

[14] Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A contrastive approach boosts the distillation of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, 2025.

[15] Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint*, arXiv:2511.10643, 2025.

[16] Yecheng Wu, Song Han, and Han Cai. Lightning OPD: Efficient post-training for large reasoning models with offline on-policy distillation. *arXiv preprint*, arXiv:2604.13010, 2026.

[17] Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggyu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. *arXiv preprint*, arXiv:2601.07155, 2026.

[18] Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint*, arXiv:2604.13016, 2026.

[19] Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint*, arXiv:2602.12275, 2026.

[20] Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint*, arXiv:2604.00626, 2026.

[21] Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: The universal logit distillation loss for LLMs. *Transactions on Machine Learning Research (TMLR)*, 2025.

[22] Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. Dual-space knowledge distillation for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*, pages 18164–18181, Miami, Florida, USA, 2024.

[23] Truong Nguyen, Phi Van Dat, Ngan Nguyen, Linh Ngo Van, Trung Le, and Thanh Hong Nguyen. CTPD: Cross tokenizer preference distillation. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, Singapore, 2026. doi: 10.1609/aaai.v40i44.41114. URL https://ojs.aaai.org/index.php/AAAI/article/view/41114.

[24] Avyav Kumar Singh, Yen-Chen Wu, Alexandru Cioba, Alberto Bernacchia, and Davide Buffelli. Cross-tokenizer LLM distillation through a byte-level interface. *arXiv preprint*, arXiv:2604.07466, 2026.

[25] Buu Phan, Ashish Khisti, and Karen Ullrich. Cross-tokenizer likelihood scoring algorithms for language model distillation. In *Proceedings of the 14th International Conference on Learning Representations (ICLR 2026)*, Rio de Janeiro, Brazil, 2026.

[26] Xiao Cui, Mo Zhu, Yulei Qin, Liang Xie, Wengang Zhou, and Houqiang Li. Multi-level optimal transport for universal cross-tokenizer knowledge distillation on language models. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI 2025)*, Philadelphia, PA, 2025.

[27] Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Unlocking on-policy distillation for any model family. *Hugging Face Blog*, 2025. https://huggingfaceh4-on-policy-distillation.hf.space/unlocking-on-policy-distillation-for-any-model-family.pdf.

[28] Xue Zhang, Songming Zhang, Yunlong Liang, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. A dual-space framework for general knowledge distillation of large language models. *arXiv preprint*, arXiv:2504.11426, 2025.

[29] Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

[30] Yijie Chen, Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. Enhancing cross-tokenizer knowledge distillation with contextual dynamical mapping. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8005–8018, Vienna, Austria, 2025.

[31] Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint*, arXiv:2603.07079, 2026.

[32] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (NeurIPS 2019)*, Vancouver, Canada, 2019.

[33] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online Event, 2020.

[34] Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

[35] Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. Rethinking kullback-leibler divergence in knowledge distillation for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025)*, 2025. URL https://aclanthology.org/2025.coling-main.383/.

[36] Hoang-Chau Luong, Dat Ba Tran, and Lingwei Chen. Diversity-aware reverse kullback-leibler divergence for large language model distillation. *arXiv preprint*, arXiv:2604.00223, 2026.

[37] Yeongmin Kim, Donghyeok Shin, Mina Kang, Byeonghu Na, and Il-Chul Moon. Distillation of large language models via concrete score matching. In *International Conference on Learning Representations*, volume arXiv:2509.25837. arXiv, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1bfc9f74afa91b9b8add5a97a97001a1-Abstract-Conference.html.

[38] Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. SelecTKD: Selective token-weighted knowledge distillation for LLMs. *arXiv preprint*, arXiv:2510.24021, 2025.

[39] Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. TIP: Token importance in on-policy distillation. *arXiv preprint*, arXiv:2604.14084, 2026.

[40] Jie Sun, Mao Zheng, Mingyang Song, Zeyuan Liu, Gengsheng Li, Houcheng Jiang, Yilin Cheng, Bichuan Feng, Yuchen Cai, Junfeng Fang, and Xiang Wang. Distill What You Trust: Reliability-Aware Multi-Teacher On-Policy Distillation, 2026. URL https://arxiv.org/abs/2609.23697.

[41] Run Yang, Runpeng Dai, Jie Sun, Jielei Zhang, Fan Zhou, Hongtu Zhu, Peiyi Li, and Longwen Gao. Influence-Directed Distillation: Solving the Diversity Bottleneck in Sampled-Token On-Policy Distillation, 2026. URL https://arxiv.org/abs/2608.29846.

[42] Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise On-policy Distillation for Small Language Model Agents, 2026. URL https://arxiv.org/abs/2605.07725.

[43] Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, and Jinqiao Wang. On-Policy Distillation with Curriculum Turn-level Guidance for Multi-turn Agents, 2026. URL https://arxiv.org/abs/2606.15912.

[44] Gengsheng Li, Mao Zheng, Mingyang Song, Jie Sun, Zeyuan Liu, Ruiqi Liu, Tianyu Yang, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Shiming Xiang, Jinqiao Wang, and Tat-Seng Chua. Data-Free On-Policy Distillation: How Far Can We Go Without External Data?, 2026. URL https://arxiv.org/abs/2609.14193.

[45] Jie Sun, Mao Zheng, Mingyang Song, Qiyong Zhong, Gengsheng Li, Zhepei Hong, Chang Wu, Pengfei Liu, Junfeng Fang, and Xiang Wang. EasyOPD: An Easy-to-use On-Policy Distillation Framework for Large Language Models, 2026. URL https://arxiv.org/abs/2607.11012.

[46] Anh Duc Le, Tu Vu, Nam Le Hai, Nguyen Thi Ngoc Diep, Linh Ngo Van, Trung Le, and Thien Huu Nguyen. CoT2Align: Cross-chain of thought distillation via optimal transport alignment for language models with different tokenizers. *arXiv preprint*, arXiv:2502.16806, 2025.

[47] Stella Eva Tsiapali, Cong-Thanh Do, and Kate Knill. Dual-space knowledge distillation with key-query matching for large language models with vocabulary mismatch. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2026)*, 2026.

[48] Haebin Shin, Lei Ji, Xiao Liu, and Yeyun Gong. Overcoming vocabulary mismatch: Vocabulary-agnostic teacher guided language modeling. *arXiv preprint*, arXiv:2503.19123, 2025.

[49] Hai An Vu, Minh-Phuc Truong, Tu Vu, and Linh Ngo Van. MoL: Mixture of layers in cross-tokenizer embedding model distillation. *Knowledge-Based Systems*, 2026.

[50] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint*, arXiv:1609.08144, 2016.

[51] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint*, arXiv:2412.15115, 2025. URL https://arxiv.org/abs/2412.15115.

[52] Microsoft, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi-ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, Min Gao, Amit Garg, Abhishek Goswami, Junheng Hao, Amr Hendy, Yuxuan Hu, Xin Jin, Mahmoud Khademi, Dongwoo Kim, Young Jin Kim, Gina Lee, Jinyu Li, Yunsheng Li, Chen Liang, Xihui Lin, Zeqi Lin, Mengchen Liu, Yang Liu, Gilsinia Lopez, Chong Luo, Piyush Madan, Vadim Mazalov, Arindam Mitra, Ali Mousavi, Anh Nguyen, Jing Pan, Daniel Perez-Becker, Jacob Platin, Thomas Portet, Kai Qiu, Bo Ren, Liliang Ren, Sambuddha Roy, Ning Shang, Yelong Shen, Saksham Singhal, Subhojit Som, Xia Song, Tetyana Sych, Praneetha Vaddamanu, Shuohang Wang, Yiming Wang, Zhenghao Wang, Haibin Wu, Haoran Xu, Weijian Xu, Yifan Yang, Ziyi Yang, Donghan Yu, Ishmam Zabir, Jianwen Zhang, Li Lyna Zhang, Yunan Zhang, and Xiren Zhou. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs. *arXiv preprint*, arXiv:2503.01743, 2025.

[53] Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint*, arXiv:2408.00118, 2024.

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

[55] Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of SLMs in grade school math. *arXiv preprint*, arXiv:2402.14830, 2024.

[56] Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. OpenMathInstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint*, arXiv:2402.10176, 2024.

[57] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

[58] Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. OpenCodeInstruct: A large-scale instruction tuning dataset for code LLMs. *arXiv preprint*, arXiv:2504.04030, 2025.

[59] Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. *arXiv preprint*, arXiv:2503.02951, 2025.

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

[61] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

[62] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

[63] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint*, arXiv:2108.07732, 2021.

[64] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*, 2025.

[65] Songming Zhang, Xue Zhang, Tong Zhang, Bojie Hu, Yufeng Chen, and Jinan Xu. KDFlow: A user-friendly and efficient knowledge distillation framework for large language models. *arXiv preprint*, arXiv:2603.01875, 2026.

[66] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias

Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint*, arXiv:2107.03374, 2021.

[67] Jie Sun, Yu Liu, Lu Han, Qiwen Deng, Xiang Shu, Yang Xiao, Lintao Ma, Xingyu Lu, Jun Zhou, Pengfei Liu, Jiancan Wu, and Xiang Wang. SepSeq: A training-free framework for long numerical sequence processing in LLMs. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 5522–5537, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.272. URL https://aclanthology.org/2026.findings-acl.272/.

[68] Jie Sun, Junkang Wu, Jiancan Wu, Zhibo Zhu, Xingyu Lu, Jun Zhou, Lintao Ma, and Xiang Wang. Robust preference optimization via dynamic target margins. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5399–5416, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.282. URL https://aclanthology.org/2025.findings-acl.282/.

[69] Yang Xiao, Mohan Jiang, Jie Sun, Keyu Li, Jifan Lin, Yumin Zhuang, Ji Zeng, Shijie Xia, Qishuo Hua, Xuefeng Li, Xiaojie Cai, Tongyu Wang, Yue Zhang, Liming Liu, Xia Wu, Jinlong Hou, Yuan Cheng, Wenjie Li, Xiang Wang, Dequan Wang, and Pengfei Liu. LIMI: Less is More for Agency, 2025. URL https://arxiv.org/abs/2509.17567.

[70] Ji Zeng, Dayuan Fu, Tiantian Mi, Yumin Zhuang, Yaxing Huang, Xuefeng Li, Lyumanshan Ye, Muhang Xie, Qishuo Hua, Zhen Huang, Mohan Jiang, Hanning Wang, Jifan Lin, Yang Xiao, Jie Sun, Yunze Wu, and Pengfei Liu. daVinci-Dev: Agent-native Mid-training for Software Engineering. In Tong Zhang, Miroslav Dudik, Martin Jaggi, Alekh Agarwal, Sharon Li, Dale Schuurmans, Jerry Zhu, Felix Berkenkamp, Hanze Dong, and Alberto Bietti, editors, *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 153265–153282. PMLR, 06–11 Jul 2026. URL https://proceedings.mlr.press/v306/zeng26d.html.

[71] Dayuan Fu, Shenyu Wu, Yunze Wu, Zerui Peng, Yaxing Huang, Jie Sun, Ji Zeng, Mohan Jiang, Lin Zhang, Yukun Li, Jiarui Hu, Liming Liu, Jinlong Hou, and Pengfei Liu. daVinci-Env: Open SWE Environment Synthesis at Scale, 2026. URL https://arxiv.org/abs/2603.13023.

[72] Dayuan Fu, Yunze Wu, Xiaojie Cai, Lyumanshan Ye, Shijie Xia, Zhen Huang, Weiye Si, Tianze Xu, Jie Sun, Keyu Li, Mohan Jiang, Junfei Wang, Qishuo Hua, Pengrui Lu, Xiangkun Hu, Yang Xiao, and Pengfei Liu. ARGO: Asynchronous Rollout with Human Guidance for Research Agent Optimization. *OpenReview preprint*, 2026. URL https://openreview.net/forum?id=JswgId3OjI.

[73] Keyu Li, Junhao Shi, Yang Xiao, Mohan Jiang, Jie Sun, Yunze Wu, Dayuan Fu, Shijie Xia, Xiaojie Cai, Tianze Xu, Weiye Si, Wenjie Li, Dequan Wang, and Pengfei Liu. AgencyBench: Benchmarking the frontiers of autonomous agents in 1M-token real-world contexts. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7422–7440, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.337. URL https://aclanthology.org/2026.acl-long.337/.

[74] Yukang Feng, Jianwen Sun, Zelai Yang, Jiaxin Ai, Chuanhao Li, Zizhen Li, Fanrui Zhang, Kang He, Rui Ma, Jifan Lin, Jie Sun, Yang Xiao, Sizhuo Zhou, Wenxiao Wu, Yiming Liu, Pengfei Liu, Shenglin Zhang, and Kaipeng Zhang. LongCLI-bench: A preliminary benchmark and study for long-horizon agentic programming in command-line interfaces. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for*

Computational Linguistics: ACL 2026, pages 29952–29963, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1497. URL https://aclanthology.org/2026.findings-acl.1497/.

[75] Yunze Wu, Dayuan Fu, Weiye Si, Zhen Huang, Mohan Jiang, Keyu Li, Shijie Xia, Jie Sun, Tianze Xu, Yang Xiao, Pengrui Lu, Xiaojie Cai, Lyumanshan Ye, Wenhong Zhu, Xiangkun Hu, and Pengfei Liu. Innovatorbench: Evaluating agents' ability to conduct innovative ai research. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 36144–36185, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3d13d910b48ac2e672a32cfdf98be1bf-Paper-Conference.pdf.

A Broader Impacts

SimCT may improve the accessibility of language-model distillation by enabling more knowledge transfer across models with different tokenizers. This can reduce the cost of training capable, smaller models and make model development more accessible. At the same time, stronger distillation can transfer undesirable teacher behaviors, including biases, unsafe responses, and hallucinations, and may lower the cost of producing models that could be misused. Therefore, SimCT should be paired with appropriate data filtering, safety evaluation, and alignment procedures before deployment.

B Training Data Construction and Curation

This appendix describes the construction of the teacher-specific cold-start SFT corpora. Within each pair and seed, the corresponding warm-start checkpoint is shared by all OPD methods. We use public math and code datasets as prompt sources, apply filtering and deduplication, generate multiple teacher responses per prompt, and select one response according to correctness, executability, and formatting criteria. The SFT training procedure that consumes this corpus is described in Appendix C.

B.1 Composition

Each teacher-specific corpus contains 10,000 examples: 5,800 mathematical reasoning examples and 4,200 code-generation examples. The math/code split roughly matches the relative emphasis of the two task families in our evaluation suite, and within each family we draw from multiple sources to avoid overfitting to any single dataset's style.

For mathematical reasoning, we sample 1,800 prompts from the GSM8K [54] training split, 2,200 from Orca-Math 200K [55], 1,000 from OpenMathInstruct-1 [56], and 800 from MATH [57], the last with the MATH-500 evaluation split [62] excluded. The MATH subset is drawn from levels 1–4 (150/150/350/150, respectively); we exclude level 5 because its solutions frequently exceed our SFT context budget and exhibit high teacher-side variance, both of which inject noise into the warm-start signal without proportionate benefit. Excluded difficulty levels are still represented in evaluation through MATH-500.

For code generation, we sample 1,800 prompts from OpenCodeInstruct [58], 900 from KodCode [59], 900 from TACO [60], and 600 from CodeContests [61]. The resulting subset covers function-level Python synthesis and competitive-programming I/O formats, with a preference toward examples that admit reliable execution-based verification.

B.2 Prompt Filtering and Deduplication

Before teacher generation, we filter prompts to remove noisy, duplicated, ill-formed, overly long, or difficult-to-verify instances, and we run overlap checks against held-out evaluation data using normalized text and near-duplicate matching to prevent test contamination.

For math sources, we retain text-only problems with clear statements, well-defined answer formats, and complete textual solutions. We discard examples with malformed questions, inconsistent answer formats, excessive derivation length, embedded code blocks, code-interpreter traces, or dependencies on diagrams, Asymptote figures, or other non-textual reasoning artifacts. We additionally apply problem-level near-duplicate filtering to suppress repeated template families.

For code sources, we keep only Python tasks with a well-defined solution target. When tests are available, we require compatibility with execution-based checking. We discard debugging, iterative-repair, SQL, shell, front-end, multi-file, interactive, project-dependent, and non-executable tasks. For contest-style sources (TACO, CodeContests), we further remove problems whose statements exceed our context budget or cannot be cleanly converted into standard Python generation examples.

B.3 Teacher Response Generation and Selection

For each retained prompt, Qwen2.5-7B-Instruct and Phi-4-mini-instruct separately sample 8 candidate responses with temperature 0.6 and top-$p = 0.95$, yielding a diverse but focused candidate set. We then select a single response per prompt as the SFT target.

Table 2: Composition of the cold-start SFT corpus. Public datasets are used as prompt sources; supervised targets are generated by Qwen2.5-7B-Instruct [51] / Phi-4-mini-instruct [52] and selected through the response-selection pipeline described in Appendix B.

Domain	Prompt source	# Examples
Math	GSM8K train [54]	1,800
	Orca-Math 200K [55]	2,200
	OpenMathInstruct-1 [56]	1,000
	MATH, excluding MATH-500 [57, 62]	800
	Math subtotal	*5,800*
Code	OpenCodeInstruct [58]	1,800
	KodCode [59]	900
	TACO [60]	900
	CodeContests [61]	600
	Code subtotal	*4,200*
	Total	**10,000**

Verification-based selection. When automatic verification is available, we keep only candidates that pass the corresponding checker. For math, we use `math_verify`[1] to compare extracted final answers under symbolic equivalence (handling fractions, equivalent algebraic forms, and standard formatting variations). For code, we execute each candidate against the dataset's provided unit tests in a sandboxed Python environment with a per-test wall-clock timeout of 10 seconds and a memory cap of 128 MB. When multiple candidates pass, we keep the shortest valid response after standardized formatting normalization, which suppresses unnecessary verbosity and limits incidental teacher-specific style.

Quality-filter selection. When verification is unavailable or partial (e.g., open-ended math without a canonical final-answer field, or code prompts without supplied tests), we apply rule-based quality filters: responses must contain a non-empty final answer in the expected format (boxed final answer for math, a complete fenced Python code block with a defined entry point for code), must not exceed 4096 tokens, and must not contain unterminated reasoning, repeated boilerplate, or empty solution bodies. The full filter rule set is documented in our released curation script. Among candidates passing all filters, we again prefer the shortest.

Role of the public datasets. Public datasets are used as *prompt sources only*; the supervised targets are generated and selected through the unified teacher-response pipeline above. This keeps warm-start supervision consistent with the teacher policy used during the subsequent OPD stage and prevents distributional drift between the two training phases. Source attribution and metadata are preserved per example throughout the pipeline.

B.4 Licenses, Attribution, and Release

Table 3 summarizes the public assets used in this work along with their roles and license terms. For each example in the released corpus we retain the source dataset, split, original identifier when available, source license, teacher model identifier, and generation configuration.

Because prompt sources carry different licenses and terms, we do not assign a single license to the constructed corpus. Per-example source attribution and license metadata are retained, and downstream users are responsible for compliance with each original asset's terms; assets with non-commercial or mixed-source terms are flagged accordingly.

The public repository, https://github.com/sunjie279/SimCT-, provides the data-preparation, teacher-generation, SFT, distillation, and evaluation workflow. The corpus can be reconstructed from the original sources using the documented pipeline; redistribution of individual examples remains subject to their source terms.

[1] https://github.com/huggingface/math-verify

Table 3: Source assets used in the main experiments. Public datasets are used as prompt sources for constructing the cold-start SFT corpus. The final supervised targets are generated by the teacher model and selected through the response-selection pipeline described in Appendix B. We preserve source attribution and license metadata for each example.

Asset	Category	Role in This Work	License / Terms
KDFlow [65]	Code	Codebase	MIT
Qwen2.5-7B-Instruct [51]	Model	Teacher model	Apache-2.0
Gemma-2-2B-IT [53]	Model	Student model	Gemma Terms of Use
Phi-4-mini-instruct [52]	Model	Teacher / student	MIT
GSM8K [54]	Math data	Prompt source	MIT
Orca-Math 200K [55]	Math data	Prompt source	MIT
OpenMathInstruct-1 [56]	Math data	Prompt source	NVIDIA License
MATH[†] [57]	Math data	Prompt source	MIT
OpenCodeInstruct [58]	Code data	Prompt source	CC BY 4.0
KodCode [59]	Code data	Prompt source	CC BY-NC 4.0
TACO[‡] [60]	Code data	Prompt source	Apache-2.0 / mixed permissive terms
CodeContests[§] [61]	Code data	Prompt source	Apache-2.0 / CC BY 4.0

[†] We use examples from the original MATH dataset [57] after excluding MATH-500 [62] evaluation instances. The source MATH dataset is released under the MIT license. [‡] TACO [60] is released under Apache-2.0 by its authors, while its official license note states that the dataset may include content under MIT and CC BY 4.0 terms. [§] CodeContests [61] licenses code under Apache-2.0 and non-code materials under CC BY 4.0; additional third-party source terms may apply.

C Training Details

For each student model, training proceeds in two stages: a warm-start SFT stage to produce a non-trivial starting policy, followed by on-policy distillation. Within each teacher–student pair and seed, all OPD methods share the same warm-start checkpoint, training prompts, and common OPD hyperparameters. Each SFT and OPD configuration is trained independently with 5 seeds; each resulting checkpoint is evaluated once. Method-specific supervision construction and loss safeguards are specified below. Hyperparameters for both stages are summarized in Table 4; this section describes each stage in detail.

C.1 Warm-Start SFT

We first SFT each student on the corresponding teacher's 10K-example corpus described in Appendix B to obtain the warm-start checkpoint from which all OPD methods start. Sharing this initialization both gives the student a non-trivial policy before OPD [11] and ensures that differences across OPD methods are not confounded by SFT data or initial checkpoints.

We optimize with AdamW using a peak learning rate of 2×10^{-6}, a linear warmup ratio of 0.05 followed by cosine decay, and train for 2 epochs. All students share the same SFT hyperparameters (Table 4). Training is performed in bfloat16 on 8×H20 96GB.

C.2 On-Policy Distillation

Starting from the warm-start checkpoint, we train each OPD method under an identical loop: at every step the student samples a prefix under its current policy, the teacher is queried on this prefix to produce next-token supervision, and the student is updated under the method-specific loss. Training prompts, prefix-generation settings (temperature 0.6, top-$p = 0.95$), optimizer, learning-rate schedule, batch construction, and training epochs are held identical across methods (Table 4).

For SimCT and SimpleOPD, the OPD loss is reverse KL [11, 12] applied in their respective supervision spaces—SimCT's space combines shared vocabulary tokens with minimal aligned units, while SimpleOPD restricts supervision to the exact teacher–student vocabulary overlap.

Normalization safeguard. All main SimCT experiments use a position-level mask on the teacher normalization multiplier. For a prefix h, define

$$Z_T(h) = \sum_{u \in \mathcal{U}_{\mathrm{SimCT}}(h)} \exp s_T(u \mid h), \qquad G(h) = Z_T(h)^{-1}, \qquad m(h) = \mathbf{1}\{G(h) \le 2\}. \quad (13)$$

A position contributes $m(h)\,\mathrm{KL}(q_S^{\mathrm{SimCT}}(\cdot \mid h) \| q_T^{\mathrm{SimCT}}(\cdot \mid h))$ to the distillation loss. Positions with $G(h) > 2$ contribute neither loss nor gradient. We apply this mask during training for all main SimCT results. The threshold is defined on the continuation-score normalizer in Eq. (8); original teacher support coverage is measured separately as in Appendix H.1. Appendix H.1 reports the multiplier distribution and the observed masking rate.

C.3 SimCT Algorithm

Algorithm 1 summarizes SimCT. Given a prompt x, the student first generates a full on-policy response y. At each prediction position t, teacher and student next-token distributions p_T^t and p_S^t are evaluated under the same prefix $(x, y_{<t})$. SimCT constructs a common supervision space $\mathcal{U}_t$ from shared vocabulary tokens augmented with minimal aligned units induced by local tokenizer-boundary mismatches, and scoring operators Π_T, Π_S use the autoregressive conditional families to produce comparable distributions q_T^t, q_S^t over $\mathcal{U}_t$ on which reverse KL is applied.

Boundary alignment and long units. The alignment is implemented on the CPU as a greedy two-pointer scan over the teacher and student token boundaries in the same underlying text. Starting from a common boundary, the scan advances through complete tokens until the next synchronized boundary, yielding the shortest span complete under both tokenizations. Each boundary is visited at most once, so alignment is linear in the number of token boundaries. Long units are retained without length-based truncation or exclusion and receive the same average log-likelihood score as shorter units. The position-level safeguard above remains applicable independently of unit length. Appendix H.2 reports the empirical length distribution, including rare long units.

Table 4: Training hyperparameters for warm-start SFT and on-policy distillation. All students share the same SFT hyperparameters; common OPD settings are shared across methods, with the SimCT-specific mask threshold listed explicitly.

	Warm-Start SFT	On-Policy Distillation
Optimizer	AdamW	AdamW
Peak learning rate	2×10^{-6}	1×10^{-6}
Warmup ratio	0.05	0.05
LR schedule	Cosine decay	Cosine decay
Weight decay	0.0	0.0
Training length	2 epochs	2 epochs
Per-device batch size	2	1
Gradient accumulation	4	8
Effective batch size	64	64
Max sequence length	4096	4096
Rollout temperature	—	0.6
Rollout top-p	—	0.95
Rollout max length	—	4096
Loss (SimCT, SimpleOPD)	—	Reverse KL
Mask cutoff $G(h)$ (SimCT only)	—	2
Precision	bfloat16	bfloat16
Hardware	8× H20	8× H20

Algorithm 1 Simple Cross-Tokenizer OPD (SimCT)

Require: Teacher $\mathcal{T}$, student $\mathcal{S}_\theta$, tokenizers τ_T, τ_S, vocabularies $\mathcal{V}_T, \mathcal{V}_S$, prompts $\mathcal{X}$, mask threshold $\gamma = 2$
Ensure: Updated student $\mathcal{S}_\theta$

for each OPD training step **do**
  Sample prompts $x \sim \mathcal{X}$
  Generate student rollouts $y \sim \mathcal{S}_\theta(\cdot \mid x)$
  Tokenize the same rollout: $\mathbf{y}^T \leftarrow \tau_T(y)$, $\mathbf{y}^S \leftarrow \tau_S(y)$
  Extract $\mathcal{A}(y)$ by a greedy scan of synchronized token boundaries
  Construct $\mathcal{U}_{\mathrm{SimCT}}(y) \leftarrow (\mathcal{V}_T \cap \mathcal{V}_S) \cup \mathcal{A}(y)$
  $\mathcal{L}_{\mathrm{SimCT}} \leftarrow 0$
  for each student-generated prefix $h = (x, y_{<t})$ **do**
    Set $\mathcal{U}_{\mathrm{SimCT}}(h) \leftarrow \mathcal{U}_{\mathrm{SimCT}}(y)$
    Compute shared-token scores $s_M(v \mid h) = \log p_M(v \mid h)$, for $M \in \{T, S\}$ and $v \in \mathcal{V}_T \cap \mathcal{V}_S$
    Score each MAU using its model-specific token realization:

$$s_M(u \mid h) = \frac{1}{k_M(u)} \sum_{j=1}^{k_M(u)} \log p_M(v_j \mid h, v_{<j}), \qquad M \in \{T, S\}.$$

    Compute $Z_T(h) \leftarrow \sum_{u \in \mathcal{U}_{\mathrm{SimCT}}(h)} \exp s_T(u \mid h)$
    Set $G(h) \leftarrow 1/Z_T(h)$
    if $G(h) \le \gamma$ **then**
      Normalize scores to obtain q_T^t, q_S^t via Eq. (8)
      $\mathcal{L}_{\mathrm{SimCT}} \leftarrow \mathcal{L}_{\mathrm{SimCT}} + \mathrm{KL}(q_S^t \| q_T^t)$
    else
      Exclude this position from the distillation loss and gradient
    end if
  end for
  Update θ using $\nabla_\theta \mathcal{L}_{\mathrm{SimCT}}$
end for
return $\mathcal{S}_\theta$

D Evaluation Benchmarks and Metrics

D.1 Benchmarks

We evaluate on four benchmarks covering complementary task regimes. **GSM8K** [54] consists of grade-school word problems requiring multi-step arithmetic from natural-language descriptions; final answers are short numerical values, but solving the problems demands a coherent reasoning trajectory. **MATH-500** [57, 62] draws from competition-style problems spanning algebra, geometry, number theory, and combinatorics; answers may be symbolic forms, fractions, or equations, making the task sensitive to both reasoning quality and output normalization. **MBPP** [63] measures function-level Python synthesis from natural-language specifications, evaluated by the dataset's provided unit tests. **LiveCodeBench-v6** [64] evaluates code generation on contemporary problems with reduced contamination, requiring more robust algorithmic reasoning under hidden tests.

These benchmarks expose tokenizer mismatch in different ways—math evaluation involves symbolic expressions and rationale tokens, while code involves identifiers, indentation, and syntax fragments—so they jointly test whether SimCT recovers useful supervision across heterogeneous output structures rather than overfitting to a single format.

D.2 Decoding

All benchmarks are evaluated zero-shot: each model receives only the task prompt, with no in-context demonstrations. This isolates capability learned during SFT and OPD from prompt-level effects, and avoids tokenizer-dependent confounds introduced by few-shot examples.

For each benchmark, we use a fixed prompt template across all methods and student models. Math prompts ask the model to solve the problem and place the final answer in `\boxed{}`; code prompts present the specification and ask for a Python solution in the benchmark's expected format. Full prompt templates are released in our supplementary code.

We decode with nucleus sampling at temperature 0.6 and top-$p = 0.95$, sampling one completion per instance. For each SFT or OPD configuration, the 5 runs correspond to independently trained checkpoints, each evaluated once. Unadapted Base models have no training stage and use 5 decoding seeds. The maximum generation length is fixed per benchmark and held identical across methods (4096 tokens for GSM8K, MATH-500 and LiveCodeBench-v6, and 2048 tokens for MBPP).

D.3 Scoring

We compute pass@1 [66] separately for each run and report the mean and standard deviation of the 5 run-level scores in Table 1. For trained models, this variability includes independent training and stochastic evaluation; it is not the standard error of the mean.

For math, we extract the final answer (preferring content inside `\boxed{}` when present) and compare against the reference using `math_verify`[2] under symbolic equivalence; this is the same checker used during SFT data curation (Appendix B), ensuring the verification standard is consistent across stages. For code, generated solutions are executed in a sandboxed Python environment with a per-test timeout of 10 seconds and a memory cap of 128 MB; we use the official MBPP test harness and the LiveCodeBench-v6 official evaluator. Postprocessing rules, the answer-extraction regex, and the sandbox configuration are identical across all compared methods.

E Theory and Proofs

This appendix provides the formal definitions and proofs for Theorem 1 and Proposition 1. The section has two goals. First, we formalize why SimCT aligned units are the minimal text units that remain unions of complete tokens under the two fixed tokenizer partitions. Second, we prove the property used in the supervision-space ablation: merging minimal aligned units removes exactly the within-span KL signal that distinguishes teacher and student preferences inside each merged span.

E.1 Tokenizer-Induced Spans and Minimal Aligned Units

Let y be a finite text span represented over a fixed atomic string representation, such as characters or bytes. We identify y with the interval $[0, |y|]$. The teacher tokenizer induces a partition

$$\mathcal{P}_T(y) = \{[b_{i-1}^T, b_i^T) : i = 1, \dots, m\}, \qquad 0 = b_0^T < \cdots < b_m^T = |y|,$$

and the student tokenizer induces a partition

$$\mathcal{P}_S(y) = \{[b_{j-1}^S, b_j^S) : j = 1, \dots, n\}, \qquad 0 = b_0^S < \cdots < b_n^S = |y|.$$

Each interval in $\mathcal{P}_T(y)$ is a complete teacher-token span, and each interval in $\mathcal{P}_S(y)$ is a complete student-token span. Throughout this proof, a jointly tokenizable unit means a union of complete consecutive tokens in each of these fixed partitions. The minimality statement is relative to these fixed partitions, with token boundaries inherited from the full text.

We first take the union of teacher and student token boundaries:

$$\mathcal{B}_\cup(y) = \mathcal{B}_T(y) \cup \mathcal{B}_S(y),$$

where $\mathcal{B}_T(y) = \{b_i^T\}_{i=0}^m$ and $\mathcal{B}_S(y) = \{b_j^S\}_{j=0}^n$. Let

$$0 = c_0 < c_1 < \cdots < c_\ell = |y|$$

be the sorted elements of $\mathcal{B}_\cup(y)$. This gives atomic fragments

$$\mathcal{F}(y) = \{[c_{k-1}, c_k) : k = 1, \dots, \ell\}.$$

These fragments are only used to define the construction. A fragment may lie inside a teacher token or a student token, and therefore is not necessarily a valid supervision unit by itself.

[2] https://github.com/huggingface/math-verify

To obtain jointly-tokenizable units, we build an undirected graph G_y whose vertices are the atomic fragments in $\mathcal{F}(y)$. Two vertices are connected if their fragments lie inside the same teacher-token span or inside the same student-token span. The connected components of G_y induce a partition of y, denoted

$$\mathcal{P}_{\mathrm{SimCT}}(y) = \{u_1, \ldots, u_r\}.$$

We call each u_k a *minimal aligned unit*. Intuitively, the connected component closes all tokenizer overlaps: if splitting a span would cut through a complete teacher token or a complete student token, the corresponding fragments must remain in the same aligned unit.

E.2 Proof of Theorem 1

Proof. We prove joint tokenizability and minimality.

Joint tokenizability. Consider any unit $u \in \mathcal{P}_{\mathrm{SimCT}}(y)$. By construction, u is a connected component of G_y. If a teacher-token span intersects u, then all atomic fragments inside that teacher token are connected in G_y, so the entire teacher-token span must lie in the same connected component. Therefore, no boundary of u cuts through a teacher token, and u is a union of consecutive complete teacher tokens.

The same argument applies to the student tokenizer. If a student-token span intersects u, all atomic fragments inside that student token are connected, so the complete student-token span lies inside u. Hence u is also a union of consecutive complete student tokens. Thus u can be represented as one or more teacher tokens and also as one or more student tokens.

Minimality. Assume for contradiction that a unit u admits a strictly finer non-empty decomposition

$$u = u' \circ u'',$$

where both u' and u'' are jointly tokenizable. Let z be the boundary between u' and u''. Since both sides are unions of complete teacher tokens, z must be a teacher-token boundary. Since both sides are also unions of complete student tokens, z must be a student-token boundary. Therefore, no teacher token or student token crosses z.

It follows that no edge in G_y can connect an atomic fragment in u' to an atomic fragment in u'', because such an edge would require the two fragments to lie inside the same teacher-token span or the same student-token span, contradicting that no token crosses z. Thus u' and u'' would belong to different connected components of G_y, contradicting that u is one connected component. Therefore, no strictly finer non-empty jointly-tokenizable decomposition exists. □

Implication. The theorem shows that SimCT does not rely on an arbitrary span length. It selects the finest units that are valid under both tokenizers. Finer atomic fragments may cut through complete tokens, while coarser mismatch spans merge multiple valid aligned units.

E.3 Proof of Proposition 1: Coarsening and Removed KL Signal

We prove Proposition 1, which is used in the supervision-space ablation. For completeness, we restate the proposition before giving the proof. Let $\mathcal{U}_{\min}$ denote the minimal aligned-unit space for a student-generated prefix h. Let

$$q_T^{\min}(\cdot \mid h), \qquad q_S^{\min}(\cdot \mid h)$$

be the teacher and student distributions over $\mathcal{U}_{\min}$. Assume $q_T^{\min}(u \mid h) > 0$ whenever $q_S^{\min}(u \mid h) > 0$, so the finite-space KL decomposition below is well defined.

A coarsening $\mathcal{C}$ partitions $\mathcal{U}_{\min}$ into disjoint coarse spans. For each coarse span $c \in \mathcal{C}$, the coarsened distributions are

$$q_T^{\mathcal{C}}(c \mid h) = \sum_{u \in c} q_T^{\min}(u \mid h), \qquad q_S^{\mathcal{C}}(c \mid h) = \sum_{u \in c} q_S^{\min}(u \mid h).$$

This is exactly the aggregation used in Eq. 11.

Proposition 1 (restated). For any coarsening $\mathcal{C}$ of the minimal aligned-unit space,

$$\mathrm{KL}\big(q_S^{\min}(\cdot \mid h) \,\|\, q_T^{\min}(\cdot \mid h)\big) \geq \mathrm{KL}\big(q_S^{\mathcal{C}}(\cdot \mid h) \,\|\, q_T^{\mathcal{C}}(\cdot \mid h)\big)\,.$$

Moreover, the difference equals the expected within-coarse-span KL:

$$\begin{aligned}&\mathrm{KL}\big(q_S^{\min}(\cdot \mid h) \,\|\, q_T^{\min}(\cdot \mid h)\big) - \mathrm{KL}\big(q_S^{\mathcal{C}}(\cdot \mid h) \,\|\, q_T^{\mathcal{C}}(\cdot \mid h)\big)\\ &\quad= \sum_{c \in \mathcal{C}} q_S^{\mathcal{C}}(c \mid h)\, \mathrm{KL}\big(q_S^{\min}(\cdot \mid h, c) \,\|\, q_T^{\min}(\cdot \mid h, c)\big)\,,\end{aligned}$$

where

$$q_S^{\min}(u \mid h, c) = \frac{q_S^{\min}(u \mid h)}{q_S^{\mathcal{C}}(c \mid h)}, \qquad q_T^{\min}(u \mid h, c) = \frac{q_T^{\min}(u \mid h)}{q_T^{\mathcal{C}}(c \mid h)}$$

for $u \in c$, with coarse spans of zero student mass omitted and the convention $0 \log 0 = 0$. Thus, coarsening removes exactly the KL signal corresponding to teacher–student discrepancies among minimal aligned units inside each merged span.

Proof. For readability, omit the conditioning on h. Let $q_S = q_S^{\min}$ and $q_T = q_T^{\min}$. For each coarse span $c \in \mathcal{C}$, write

$$Q_S(c) = \sum_{u \in c} q_S(u), \qquad Q_T(c) = \sum_{u \in c} q_T(u).$$

Then $Q_S = q_S^{\mathcal{C}}$ and $Q_T = q_T^{\mathcal{C}}$.

We decompose the minimal-unit KL by grouping terms according to their coarse span:

$$\mathrm{KL}(q_S \| q_T) = \sum_{c \in \mathcal{C}} \sum_{u \in c} q_S(u) \log \frac{q_S(u)}{q_T(u)}.$$

For any $u \in c$, write

$$q_S(u) = Q_S(c)\, q_S(u \mid c), \qquad q_T(u) = Q_T(c)\, q_T(u \mid c),$$

where

$$q_S(u \mid c) = \frac{q_S(u)}{Q_S(c)}, \qquad q_T(u \mid c) = \frac{q_T(u)}{Q_T(c)}.$$

Substituting this factorization gives

$$\begin{aligned}\mathrm{KL}(q_S \| q_T) &= \sum_{c \in \mathcal{C}} \sum_{u \in c} Q_S(c) q_S(u \mid c) \log \frac{Q_S(c) q_S(u \mid c)}{Q_T(c) q_T(u \mid c)}\\ &= \sum_{c \in \mathcal{C}} Q_S(c) \log \frac{Q_S(c)}{Q_T(c)} + \sum_{c \in \mathcal{C}} Q_S(c) \sum_{u \in c} q_S(u \mid c) \log \frac{q_S(u \mid c)}{q_T(u \mid c)}\\ &= \mathrm{KL}(Q_S \| Q_T) + \sum_{c \in \mathcal{C}} Q_S(c)\, \mathrm{KL}(q_S(\cdot \mid c) \,\|\, q_T(\cdot \mid c))\,.\end{aligned}$$

Since the second term is a nonnegative weighted sum of KL divergences,

$$\mathrm{KL}(q_S \| q_T) \geq \mathrm{KL}(Q_S \| Q_T).$$

Rearranging the equality above gives the stated expression for the removed KL signal. □

Connection to the ablation. Equation 12 averages the difference above over student-generated prefixes:

$$\Delta_{\mathcal{C}} = \mathbb{E}_h \left[\mathrm{KL}(q_S^{\min}(\cdot \mid h) \| q_T^{\min}(\cdot \mid h)) - \mathrm{KL}(q_S^{\mathcal{C}}(\cdot \mid h) \| q_T^{\mathcal{C}}(\cdot \mid h))\right].$$

By the proposition, $\Delta_{\mathcal{C}}$ is nonnegative and measures the expected within-span KL signal erased by the coarsening $\mathcal{C}$. The right panel of Figure 4 therefore directly tests the empirical hypothesis motivated by the theory: removing more within-span teacher–student discrepancy may reduce downstream gains. Here the KL decomposition quantifies removed distributional signal, while the downstream relationship is evaluated empirically.

Table 5: **Training time comparison.** We report wall-clock training time on a single machine with 8 NVIDIA H20 GPUs. All methods use the same training epochs, hardware, and common protocol; wall-clock time is measured rather than fixed.

Method	**Qwen2.5→Gemma2**	**Phi4→Gemma2**	**Qwen2.5→Phi4**	**Average**
SimpleOPD	2.48h	2.61h	2.50h	2.53h
SimCT	2.70h	2.73h	2.64h	2.69h
ALM	2.84h	2.88h	2.79h	2.84h
GOLD	2.93h	2.89h	2.97h	2.93h
DSKD	2.90h	2.96h	2.91h	2.92h

Table 6: Estimated compute resources for reproducing the reported experiments. All runs use a single machine with $8\times$ NVIDIA H20 96GB GPUs. Training configurations use 5 seeds, with one evaluation per trained checkpoint; Base models use 5 decoding seeds. Cost is reported in days on the $8\times$H20 server, computed as (number of runs $\times$ wall-clock hours per run) / 24.

Stage	Component	Runs	Runtime per run	$8\times$H20 days
Cold start	Qwen2.5 response generation	1×5	1 h	0.21
	Phi-4 response generation	1×5	1 h	0.21
	Phi-4 SFT (Qwen-generated data)	1×5	1 h	0.21
	Gemma SFT (Qwen-generated data)	1×5	1 h	0.21
	Gemma SFT (Phi-generated data)	1×5	1 h	0.21
	Cold-start subtotal	25	—	*1.04*
Training	Qwen $\rightarrow$ Phi, five methods	5×5	3 h	3.13
	Qwen $\rightarrow$ Gemma, five methods	5×5	3 h	3.13
	Phi $\rightarrow$ Gemma, five methods	5×5	3 h	3.13
	Training subtotal	75	—	*9.38*
Evaluation	Base models (Qwen, Phi, Gemma)	3×5	1 h	0.63
	Warm-start (3 pairs)	3×5	1 h	0.63
	Qwen $\rightarrow$ Phi (5 methods)	5×5	1 h	1.04
	Qwen $\rightarrow$ Gemma (5 methods)	5×5	1 h	1.04
	Phi $\rightarrow$ Gemma (5 methods)	5×5	1 h	1.04
	Evaluation subtotal	105	—	*4.38*
			Total	**14.79**

E.4 Discussion

The theorem and proposition play complementary roles. The theorem identifies the finest jointly-tokenizable units available under the two tokenizer partitions. The proposition shows what is lost when those units are merged: coarsening preserves the total probability of each merged span, but discards the conditional teacher–student differences among the minimal units inside it.

These results rely only on tokenizer boundaries and probability aggregation. They do not assume a learned semantic alignment, an embedding similarity metric, or an optimal-transport plan. This matches the design of SimCT: the method changes the supervision space to the finest comparable text units while keeping the OPD objective itself unchanged.

F Computational Resources

All experiments were conducted on a single server equipped with dual AMD EPYC 9K84 CPUs, 384 logical CPU cores, 2.2TB system memory, and $8\times$ NVIDIA H20 GPUs with approximately 96GB memory per GPU. Table 5 reports the measured wall-clock time of a single cross-tokenizer OPD run under this hardware setting. Across teacher–student pairs and methods, one training run typically takes about 2.5–3 hours. Averaged across the reported pairs, SimpleOPD takes 2.53 hours and SimCT takes 2.69 hours, an increase of 0.16 hours (approximately 6.3%). These totals cover the

full OPD training run, including student rollouts, teacher forward passes, alignment, and optimization. Boundary alignment is a linear CPU scan as described in Appendix C. SimCT is also faster than ALM, GOLD, and DSKD across the reported settings.

Table 6 summarizes the compute required to reproduce the reported main experiments. Each SFT and cross-tokenizer OPD configuration is trained independently with 5 seeds. Each resulting checkpoint is evaluated once, and Table 1 aggregates the resulting run-level scores. The cold-start accounting also includes the teacher-generation runs listed in Table 6. For the aggregate estimate, we use a conservative rounded runtime of approximately 3 hours per cross-tokenizer OPD run, consistent with the measured runtimes in Table 5. The total cost is approximately 14.8 days on a single 8× H20 server. The total consists of three parts. First, the warm-start stage costs about 1.0 day on a single 8× H20 server, covering two teacher response-generation runs and three student SFT runs (Phi-4 on Qwen-generated data, Gemma-2 on Qwen-generated data, and Gemma-2 on Phi-generated data), each repeated over 5 random seeds. Second, cross-tokenizer OPD training costs about 9.4 days on a single 8× H20 server, covering three teacher–student pairs (Qwen2.5→Phi4, Qwen2.5→Gemma-2, and Phi4→Gemma-2), each trained with five methods (SimpleOPD, SimCT, ALM, GOLD, and DSKD), and each method-pair configuration repeated over 5 random seeds, yielding $3 \times 5 \times 5 = 75$ training runs. Third, evaluation costs about 4.4 days on a single 8× H20 server, covering 3×5 Base-model decoding runs, 3×5 independently trained warm-start checkpoints, and 15×5 independently trained OPD checkpoints. Each trained checkpoint is evaluated once, yielding $15 + 15 + 75 = 105$ evaluation runs.

The reported total only accounts for the runs needed to reproduce the main experimental results. It excludes preliminary studies, hyperparameter exploration, debugging runs, ablation studies in Section 4.3 and Section 4.4, the additional analyses in Appendix H, and failed or repeated jobs during development. The full project compute is therefore larger than the reproducibility cost reported here.

Finally, the end-to-end wall-clock cost of cross-tokenizer OPD is dominated by student rollouts and teacher forward passes. SimCT operates at the distillation-interface level. It introduces no trainable projection module, requires no global vocabulary-level alignment, and keeps the standard OPD training pipeline unchanged. As a result, SimCT remains close to the lightweight SimpleOPD baseline and is faster than more complex cross-tokenizer baselines, while recovering substantially more teacher supervision than SimpleOPD.

G Case Study

We provide qualitative case studies on math and code examples to illustrate how different supervision interfaces can affect cross-tokenizer OPD behavior at the local level. For each example, we compare SimCT with representative baselines under the same teacher–student setting and report the final answer or execution result. These cases highlight representative local errors—an incorrect modular computation, a missing symmetry factor, a mismatched input-output convention, or an invalid function signature—that are easy to miss in aggregate scores. Several baselines recover the high-level solution template but fail on these local distinctions, which is consistent with the supervision loss caused by tokenizer mismatch as measured in Section 4.3. SimCT avoids these specific errors in the shown examples, providing qualitative intuition for how finer-grained supervision can help preserve local reasoning accuracy. These examples complement the aggregate evaluation in Table 1.

Case Study 1: Number Theory (Level 4) — Qwen2.5 → Gemma-2-2B *MATH500*

Question: Find the modulo 7 remainder of the sum $1 + 3 + 5 + 7 + 9 + \cdots + 195 + 197 + 199$.
Ground-truth Answer: 4

Method	Response (Abbreviated)	Result
SimCT (Ours)	*The sequence* $1, 3, 5, \ldots, 199$ *has* $n = 100$ *terms. Sum* $= \frac{100}{2}(1 + 199) = 10000$. *Since* $10000 = 7 \times 1428 + 4$, *the remainder is* 4. **Answer:** $\boxed{4}$	✓ **Correct**
SimpleOPD	*Correctly identifies 100 terms and computes the sum as 10000. However, makes an arithmetic error in the final modular division step, computing* $10000 \div 7 = 1428$ *remainder* 3 *instead of* 4. **Answer:** 3	✗ **Incorrect**
ALM	*Incorrectly counts the number of terms as 199 (confusing the last term with the count). Computes a wrong sum and obtains remainder* 1. **Answer:** 1	✗ **Incorrect**
DSKD	*Correctly sets up the arithmetic sequence formula but makes an error in computing* $10000 \bmod 7$, *writing the valid identity* $10000 = 7 \times 1429 - 3$ *but reporting the incorrect nonnegative remainder* 2. **Answer:** 2	✗ **Incorrect**
GOLD	*Identifies the correct sum of 10000 but then incorrectly applies modular arithmetic, computing* $10000 \bmod 7 = 3$ *(same error as SimpleOPD).* **Answer:** 3	✗ **Incorrect**

Analysis

SimCT correctly counts 100 terms, obtains the sum 10000, and computes $10000 = 7 \times 1428 + 4$. ALM already errs in counting the terms; SimpleOPD, DSKD, and GOLD fail at the remainder calculation. These responses illustrate distinct local reasoning errors.

Case Study 2: Counting & Probability (Level 4) — Qwen2.5 → Phi4-mini *MATH500*

Question: In how many ways can 8 people be seated around a square table with 2 people on a side? (Two configurations are considered equivalent if one is a rotation of another.)
Ground-truth Answer: 10080

Method	Response (Abbreviated)	Result
SimCT (Ours)	*Total linear arrangements:* $8! = 40320$. *A square table has 4 rotational symmetries (0°, 90°, 180°, 270°). Dividing:* $8!/4 = 40320/4 = 10080$. **Answer:** $\boxed{10080}$	✓ **Correct**
SimpleOPD	*Attempts Burnside's lemma but incorrectly computes* $\mathrm{Fix}(180°) = 4! \times 2^4 = 384$ *instead of 0, yielding* $\frac{40320+384}{4} = 10176$. **Answer:** 10176	✗ **Incorrect**
ALM	*Incorrectly treats the problem as a circular permutation first* $(8!/8 = 7!)$*, then divides by 4 again for square symmetry, double-counting:* $7!/4 = 1260$. **Answer:** 1260	✗ **Incorrect**
DSKD	*Fixes one person at a corner, then incorrectly partitions the remaining 7 people into "pairs" using* $\frac{7!}{(2!)^3} = 630$*, conflating seating with pairing.* **Answer:** 630	✗ **Incorrect**
GOLD	*Fixes one person, computes* $7!$*, then multiplies by* $(2!)^4 = 16$ *(incorrectly adding intra-side permutations already counted), yielding* $\frac{7!\times 16}{4} = 20160$. **Answer:** 20160	✗ **Incorrect**

Analysis

SimCT applies the clean and correct approach: total permutations divided by rotational symmetries $(8!/4)$. Each baseline attempts a different—and incorrect—strategy: SimpleOPD misapplies Burnside's lemma; ALM double-divides by both circular and square symmetries; DSKD confuses seating with pairing; and GOLD redundantly multiplies by intra-side arrangements. The shown responses differ in how they handle rotational equivalence; SimCT gives the correct count in this example.

Case Study 3: Code Generation — Qwen2.5 → Gemma-2-2B *LiveCodeBench*

Problem (abc389_b): You are given an integer X not less than 2. Find the positive integer N such that $N! = X$. It is guaranteed that there is exactly one such N.
Constraints: $2 \leq X \leq 3 \times 10^{18}$

Method	Generated Code (Key Logic)	Tests	Result

SimCT (Ours)	`def findN(x):`
	`  n = 1`
	`  fact = 1`
	`  while fact < x:`
	`  n += 1`
	`  fact *= n`
	`  return n`
	`x = int(input().strip())`
	`print(findN(x))`
		42/42	✓ Pass

SimpleOPD	`def find_n():`
	`  for i in range(2, int(1e18) + 1):`
	`  if factorial(i) == int(input()):`
	`  return i`
	▷ Calls `input()` again if the first candidate fails
		3/42	✗ Fail

ALM	`def find_N():`
	`  X = int(input())`
	`  N = 1`
	`  while True:`
	`  N = factorial(N)  # overwrites N!`
	`  if N == X: return N`
	`  else: N += 1`
	▷ Overwrites the index N with $N!$ and can return X instead of N
		0/42	✗ Fail

DSKD	`# Outputs explanation text before code`
	`def find_n():`
	`  X = int(input())`
	`  for n in range(2, X + 1):`
	`  if factorial(n) == X: return n`
		0/42	✗ Fail

GOLD	`def find_n():`
	`  for n in range(2, 3*10**18 + 1):`
	`  if factorial(n) == int(input()):`
	`  return n`
	▷ Calls `input()` again if the first candidate fails
		3/42	✗ Fail

Analysis

SimCT reads the input once and uses $O(N)$ multiplications to reach $X = N!$; the constraints imply $N \leq 20$. SimpleOPD and GOLD reread input inside the loop, whereas ALM overwrites the search index with a factorial. Test counts refer to the complete evaluated outputs.

Case Study 4: Code Generation — Qwen2.5 → Phi4-mini *MBPP*

Problem: Write a python function to find the largest number that can be formed with the given digits.
Test Cases:

```
assert find_Max_Num([1,2,3], 3) == 321
assert find_Max_Num([4,5,6,1], 4) == 6541
assert find_Max_Num([1,2,3,9], 4) == 9321
```

Method	Generated Code (Key Logic)	Result

SimCT (Ours)	`def find_Max_Num(digits, n):`
	`  digits.sort(reverse=True)`
	`  return int(''.join(map(str, digits[:n])))`
	▷ Correct signature `(digits, n)` matching test cases
		✓ Pass

SimpleOPD	`def find_Max_Num(digits):`
	`  sorted_digits = sorted(digits, reverse=True)`
	`  return int(''.join(map(str, sorted_digits)))`
	▷ Missing parameter `n`; call `find_Max_Num([1,2,3], 3)` raises `TypeError`
		✗ Fail

ALM	`def find_Max_Num(digits):`
	`  digits.sort(reverse=True)`
	`  return int(''.join(map(str, digits)))`
	▷ Missing parameter `n`; same `TypeError` on test invocation
		✗ Fail

DSKD	`def find_Max_Num(digits):`
	`  return int(''.join(sorted(map(str,digits), reverse=True)))`
	▷ Missing parameter `n`; the two-argument test call raises `TypeError`
		✗ Fail

GOLD	`def find_Max_Num(digits):`
	`  digits.sort(reverse=True)`
	`  return int(''.join(map(str, digits)))`
	▷ Missing parameter `n`; identical failure mode as ALM
		✗ Fail

Analysis

This example reveals a subtle but critical failure: all baseline methods generate *functionally correct* sorting logic, but define the function with only one parameter `(digits)` instead of the required two-parameter signature `(digits, n)`. When the test harness calls `find_Max_Num([1,2,3], 3)`, all baselines raise a `TypeError` due to the unexpected second argument. SimCT supplies the required two-argument signature and passes the displayed tests. The baseline outputs contain function-interface errors.

H Additional Analyses of Recovered Supervision

This section examines which teacher predictions are represented by the candidate space, what the recovered units look like, and where their supervision helps. We distinguish original next-token support coverage from the normalized continuation scores used for optimization.

H.1 Candidate-Space Coverage and Normalization

Support coverage versus score normalization. Let $\mathcal{R}_T(h) \subseteq \mathcal{V}_T$ denote the set of original teacher token identities represented by shared tokens or by the teacher-side realizations of aligned MAUs at prefix h. A teacher token is counted once in this set even if it is represented more than once. Define

$$M(h) = \sum_{v \in \mathcal{R}_T(h)} p_T(v \mid h), \qquad r(h) = 1 - M(h). \tag{14}$$

This diagnostic sums probabilities in the original teacher vocabulary. In particular, $\mathcal{R}_T(h)$ is a set of teacher tokens, whereas $\mathcal{U}_{\mathrm{SimCT}}(h)$ contains shared tokens and potentially multi-token continuation units. The coverage statistic $M(h)$ and the continuation-score normalizer $Z_T(h)$ in Eq. (13) therefore measure different quantities.

Table 7 reports measurements over 263,817 on-policy rollout positions. Mean support coverage is 99.5%, and the overall 95th percentile of $G(h)$ is 1.008. The 1.13% of positions with $G(h) > 2$ are excluded from the distillation objective by the training rule in Appendix C.2.

Table 7: Original teacher support coverage and the actual score-normalization multiplier. P_5 and P_{95} denote percentiles. All mass and frequency columns are percentages; G is dimensionless.

Pair	Positions	Mean M	$P_5(M)$	$M < .95$	$P_{95}(G)$	$G > 2$
Qwen → Phi	59,585	99.9	100.0	0.1	1.000	0.11
Qwen → Gemma	110,200	99.3	99.9	1.6	1.018	1.62
Phi → Gemma	94,032	99.4	99.7	1.6	1.026	1.20
Overall	263,817	99.5	99.9	1.3	1.008	1.13

For a retained shared token v, the score remains $\log p_T(v \mid h)$, so

$$q_T^{\mathrm{SimCT}}(v \mid h) = G(h) p_T(v \mid h), \qquad \frac{q_T^{\mathrm{SimCT}}(v_i \mid h)}{q_T^{\mathrm{SimCT}}(v_j \mid h)} = \frac{p_T(v_i \mid h)}{p_T(v_j \mid h)}. \tag{15}$$

Thus, relative rankings and probability ratios are preserved *among shared tokens*. The reported 95th percentile places the multiplier $G(h)$ at or below 1.008 for at least 95% of measured positions, bounding upward rescaling at those positions. For an MAU, $\exp s_T(u \mid h)$ is a geometric mean of conditional token probabilities under the teacher's realization. Support coverage $M(h)$ summarizes represented mass in the original teacher vocabulary, while q_T^{SimCT} normalizes these model-specific continuation scores over the candidate space.

Recovered and residual support. Table 8 compares exact shared-token support with the support represented after adding MAUs. The overall mean increases from 95.6% to 99.5%. The mean residual is 0.5%; Tables 10 and 11 further characterize its distribution and the high-probability predictions it contains.

Table 8: Mean original teacher probability mass (%). Residual mass is measured outside $\mathcal{R}_T(h)$.

Pair	Shared tokens	With MAUs	Residual
Qwen → Phi	99.3	99.9	0.1
Qwen → Gemma	94.8	99.3	0.7
Phi → Gemma	94.0	99.4	0.6
Overall	95.6	99.5	0.5

High-probability predictions and tail cases. Let $\mathcal{K}_T(h)$ be the teacher's top-K next-token set. Probability-weighted recall is

$$\text{Recall}\,@K(h) = \frac{\sum_{v \in \mathcal{K}_T(h) \cap \mathcal{R}_T(h)} p_T(v \mid h)}{\sum_{v \in \mathcal{K}_T(h)} p_T(v \mid h)}. \tag{16}$$

The complementary quantity $\text{MassOutside}\,@K(h) = 1 - \text{Recall}\,@K(h)$ measures the fraction of top-K probability mass outside the represented support. Table 9 summarizes recall across positions. Tables 10 and 11 separate math and code and expose tail cases that an overall mean can hide.

Table 9: Probability-weighted top-K recall across rollout positions (fractions, rounded to three decimal places).

Statistic	$K = 1$	$K = 5$	$K = 20$	$K = 100$
Mean	0.995	0.995	0.995	0.995
Median	1.000	1.000	1.000	1.000
5th percentile	1.000	1.000	0.999	0.999

Table 10: Fraction of positions (%) whose residual original teacher probability mass exceeds each threshold.

Domain	$r(h) > 5\%$	$r(h) > 10\%$	$r(h) > 20\%$	$r(h) > 50\%$
Math	1.49	1.13	0.82	0.45
Code	1.06	0.82	0.63	0.39
Overall	1.27	0.97	0.72	0.42

Table 11: Mean top-K probability mass outside the represented support, expressed as a percentage of each position's top-K mass.

Domain	Top-1	Top-5	Top-20	Top-100
Math	0.52	0.59	0.60	0.61
Code	0.42	0.47	0.47	0.47
Overall	0.47	0.53	0.54	0.54

Residual mass exceeds 50% at 0.42% of measured positions. The omitted support also occasionally includes the teacher's top-1 prediction. The $G(h) > 2$ mask selects positions by continuation-score amplification, while residual mass and top-K recall describe coverage in the original teacher vocabulary. Together, these diagnostics characterize both the typical coverage and the observed tail cases.

H.2 MAU Lengths and Surface Categories

We extracted 30,215 recovered MAUs from 900 on-policy responses. Table 12 reports token and character lengths. Most units are short: the overall median is two tokens for either model, and four characters. Rare long spans arise when the two tokenizations do not resynchronize over structured strings. The longest observed unit is a repeated `.invalid.test` string with 447 characters, 70 teacher tokens, and 140 student tokens.

Table 12: MAU length statistics. Each length entry is mean / median / 95th percentile / maximum. Character lengths are reported separately from model-specific token counts.

Pair	MAUs	Teacher tokens	Student tokens	Characters
Qwen → Phi	4,647	2.16 / 2 / 3 / 18	1.31 / 1 / 2 / 6	3.97 / 3 / 10 / 26
Qwen → Gemma	11,568	2.24 / 2 / 5 / 70	3.12 / 2 / 8 / 140	8.83 / 6 / 24 / 447
Phi → Gemma	14,000	1.88 / 2 / 4 / 22	2.70 / 2 / 6 / 31	6.87 / 4 / 21 / 156
Overall	30,215	2.06 / 2 / 4 / 70	2.65 / 2 / 6 / 140	7.17 / 4 / 21 / 447

Long MAUs are neither truncated nor discarded solely because of their length. As in Eq. (7), they use the average log-likelihood

$$s_M(u \mid h) = \frac{1}{k_M(u)} \sum_{j=1}^{k_M(u)} \log p_M(v_j \mid h, v_{<j}) = \log \left(\prod_{j=1}^{k_M(u)} p_M(v_j \mid h, v_{<j}) \right)^{1/k_M(u)}.$$

This avoids accumulating negative log-probability merely because a realization contains more tokens. The score remains tied to each model's tokenization through its conditional factors and token count $k_M(u)$. The independent position mask in Appendix C.2 still applies.

Surface forms. Table 13 groups the recovered units by their surface forms and gives representative segmentations. Words or word-pieces, operators or punctuation, code expressions, and numbers together account for 97.1% of the units. These categories describe the observed corpus rather than a semantic requirement of the MAU construction: a minimal boundary-consistent unit need not be a complete word or expression.

Table 13: Surface categories of the 30,215 recovered MAUs. Brackets delimit individual tokens in the illustrative examples; a dash means that no example is displayed. Percentages sum to 99.9% because of rounding.

Category	%	Teacher split	Student split
Word / word-piece	29.4	`[Let][’s]`	`[Let’s]`
Operator / punctuation	25.9	`[\)]`	`[\][()]`
Code token / expression	21.3	`[/][week]`	`[/week]`
Number / numeric	20.5	`[2][0]`	`[20]`
LaTeX / math markup	1.3	`[{T]`	`[{][T]`
Multi-word phrase	0.7	—	—
Alphanumeric (var./coef.)	0.2	—	—
Whitespace / newline	0.6	—	—

For example, `Here’s` is realized as `[Here’s]` by the teacher and `[Here][’][s]` by the student in one recovered unit. Another code span, `_sequences(n,` is realized as `[_sequences][(n]` versus `[_][sequences][(][n]`. These examples illustrate how a common surface span can preserve supervision when its constituent tokens cannot be compared one to one.

H.3 Controlled Overlap and Same-Tokenizer References

Controlled effective vocabulary overlap. To vary overlap while keeping the models fixed, we use VibeThinker-3B as teacher and Qwen2.5-Coder-3B-Instruct as student, which share a tokenizer. We construct nested effective shared vocabularies at 85%, 70%, and 55% overlap, representing removed tokens with synthetic MAUs following the empirical MAU length distribution. Models, data, and training are held fixed. This intervention restricts the supervision interface; it does not change either pretrained tokenizer. Full-vocabulary OPD on the same model pair provides an empirical reference with an average score of 52.89.

Let S_{shared}, S_{SimCT}, and S_{full} denote the average scores of SimpleOPD, SimCT, and full-vocabulary OPD. We report the fraction of the reference gap closed:

$$R_{\text{gap}} = \frac{S_{\text{SimCT}} - S_{\text{shared}}}{S_{\text{full}} - S_{\text{shared}}}. \tag{17}$$

Table 14 averages GSM8K, MATH500, MBPP, and LiveCodeBench. As effective overlap decreases, SimCT's absolute gain over SimpleOPD increases from 3.44 to 4.71 points, while R_{gap} decreases from 81.7% to 74.5%. The full-vocabulary reference remains fixed across these settings. This empirical reference measures the effect of restricting the supervision interface under a fixed shared tokenizer.

Table 14: Controlled effective overlap with a shared-tokenizer model pair. Gain is SimCT minus SimpleOPD in score points; R_{gap} is defined in Eq. (17).

Overlap	SimpleOPD	SimCT	Full vocabulary	Gain	R_{gap}
85%	48.68	52.12	52.89	3.44	81.7%
70%	47.53	51.67	52.89	4.14	77.2%
55%	46.57	51.28	52.89	4.71	74.5%

Reference under natural tokenizer mismatch. We additionally compare Phi-4-mini $\rightarrow$ Gemma-2-2B with a same-tokenizer Gemma-2-9B $\rightarrow$ Gemma-2-2B full-vocabulary reference, holding the student initialization, data, and training hyperparameters fixed. We use the instruction-tuned checkpoints (Gemma-2-2B-it, Gemma-2-9B-it, and Phi-4-mini-instruct); Table 15 includes the base student and both teachers for context. SimCT improves over SimpleOPD on each of the four benchmarks and by 1.09 points on average (32.58 versus 31.49).

Table 15: Natural mismatch and a same-tokenizer reference. The two settings use Phi-4-mini and Gemma-2-9B teachers, respectively. LCB denotes LiveCodeBench.

Model / distillation setting	GSM8K	MATH500	MBPP	LCB	Average
Base Gemma-2-2B	61.85	17.96	36.92	6.65	30.85
Teacher Gemma-2-9B	88.25	43.80	57.00	10.29	49.84
Teacher Phi-4-mini	85.75	54.56	55.36	7.92	50.90
Phi $\rightarrow$ Gemma, SimpleOPD	62.62	19.00	37.89	6.43	31.49
Phi $\rightarrow$ Gemma, SimCT	62.92	21.08	39.16	7.14	32.58
Gemma $\rightarrow$ Gemma, full vocab.	65.04	22.16	40.52	7.48	33.80

Using the base student's score $S_{\text{base}} = 30.85$ as the denominator anchor gives a separate descriptive quantity,

$$R_{\text{base}}(S) = \frac{S - S_{\text{base}}}{33.80 - S_{\text{base}}}. \tag{18}$$

It is approximately 59% for SimCT and 22% for SimpleOPD. Here "base" denotes the unadapted Gemma-2-2B student. The fractions quantify progress toward the gain of the Gemma-2-9B full-vocabulary reference; Eq. (17) separately measures recovery in the fixed-teacher controlled-overlap experiment.

Gains relative to the OPD starting point. For the main experiments in Table 1, comparison with the actual SFT warm-start gives average-score gains of 1.86 points (30.94 → 32.80, approximately 6.0%) for Qwen → Gemma, 3.76 points (51.30 → 55.06, 7.3%) for Qwen → Phi, and 1.47 points (31.11 → 32.58, 4.7%) for Phi → Gemma. These improvements use the SFT warm-start as their baseline.

H.4 Within-Unit Scoring Normalization

On Qwen2.5 → Gemma-2, we compare three within-MAU scoring rules under otherwise identical training. Write the unnormalized continuation log-likelihood as

$$\ell_M(u \mid h) = \sum_{j=1}^{k_M(u)} \log p_M(v_j \mid h, v_{<j}).$$

The alternatives are no length normalization, $s_M = \ell_M$; token-length normalization, $s_M = \ell_M / k_M(u)$; and byte-length normalization, $s_M = \ell_M / b(u)$, where $b(u)$ is the byte length of the unit's surface text. Each variant subsequently normalizes candidate scores as in Eq. (8).

Table 16: Within-unit normalization ablation on Qwen2.5 → Gemma-2. Scores are reported to one decimal place; LCB denotes LiveCodeBench.

Scoring rule	GSM8K	MATH500	MBPP	LCB	Average
No normalization	62.3	21.6	37.4	6.5	32.0
Token-length (SimCT)	**63.5**	**22.2**	**38.2**	**7.3**	**32.8**
Byte-length	62.9	21.5	37.6	6.9	32.2

Token-length normalization gives the highest reported score on each benchmark, with an average gain of 0.8 points over no normalization and 0.6 points over byte-length normalization. Averaging log-probabilities avoids a direct penalty from the number of factors in a unit's realization, while preserving its model-specific token factorization. These results support the practical choice in Eq. (7) for this model pair.

H.5 Position-Level Supervision Analysis

To localize the benefit of recovered supervision, we perform a controlled ablation on Qwen2.5 → Gemma-2. Each variant supervises only the indicated MAU subset. The budget is the fraction of available MAUs used in this ablation; compared subsets are matched by count. Random subsets provide controls at the same budget. We distinguish selection by teacher next-token entropy, by shared-vocabulary probability mass, and by surface content (numbers, code expressions, and operators versus word-pieces or formatting).

Let S_{variant} be the score of a selection variant and $S_{\text{Full SimCT}}$ the score when all MAUs are used. The reported recovery is

$$R_{\text{subset}} = \frac{S_{\text{variant}} - S_{\text{SimpleOPD}}}{S_{\text{Full SimCT}} - S_{\text{SimpleOPD}}}. \tag{19}$$

Pure SimpleOPD defines 0% recovery and Full SimCT defines 100%. The no-MAU anchor includes the out-of-shared-vocabulary (OoSV) supervision used in Section 4.3; it is therefore distinct from pure SimpleOPD and achieves 18% recovery.

High- and low-entropy selection give similar recovery (76% and 74%), close to the 75% random control. Mean teacher entropy is also similar at MAU and aligned positions (0.21 and 0.23 nats). These measurements do not indicate that the benefit is concentrated at high-entropy positions. In contrast, low-shared-mass selection outperforms both high-shared-mass selection and the matched random control (88% versus 56% and 75%). Content selection also outperforms other surfaces and its random control (83% versus 45% and 66%). Within this experiment, recovered supervision is especially useful where shared-token supervision is weak and for arithmetic or code-related surfaces.

Table 17: Position-level ablation with matched MAU budgets. Recovery is R_{subset} from Eq. (19). The 37% budget is the matched selection ratio in this experiment.

Variant	MAU positions supervised	Budget	Recovery
No-MAU (SimpleOPD + OoSV)	None	0%	18%
Full SimCT	All	100%	100%
Random-50	Random	50%	75%
High-Entropy-50	Highest teacher entropy	50%	76%
Low-Entropy-50	Lowest teacher entropy	50%	74%
Low-SharedMass-50	Lowest shared-vocabulary mass	50%	88%
High-SharedMass-50	Highest shared-vocabulary mass	50%	56%
Content-Matched	Numbers / code / operators	37%	83%
Random-Matched	Random, count-matched	37%	66%
Other-Matched	Word-pieces / formatting	37%	45%

H.6 Relation to Prior Methods and Off-Policy Applicability

Scope of the comparison. Cross-tokenizer methods differ in both their alignment mechanisms and the objects they compare. Table 18 summarizes the relevant distinctions.

Table 18: Comparison of the supervision or scoring interfaces. Entries summarize alignment mechanisms and scoring objects.

Method	Alignment and comparison
GOLD [27]	Aligns token sequences through probability merging; its hybrid vocabulary loss combines direct shared-token mapping with a sorted-probability loss for unmatched tokens.
ALM [29]	Compares likelihoods of aligned chunks through binarized divergences over a chunk and its complement; it also considers hidden-state distillation.
Cross-tokenizer likelihood scoring [25]	Uses BPE structure to evaluate a model's likelihoods under another tokenizer; provides exact special cases, a general lossless procedure, and faster approximations.
SimCT	Constructs a shared-token-plus-MAU candidate space, assigns normalized continuation scores, and applies reverse KL without a trainable projection module.

SimCT's theoretical minimality concerns synchronized boundaries of the two fixed tokenizations (Appendix E). The coarsening ablation compares alternative granularities within this construction. SimCT uses normalized continuation scores, while the likelihood-scoring method targets cross-tokenizer likelihood evaluation. Runtime comparisons in Appendix F apply to the evaluated implementations and settings.

Applicability beyond on-policy prefixes. Constructing aligned units and scoring their continuations does not intrinsically require prefixes to be sampled from the current student. The same interface can in principle be evaluated on fixed-corpus or teacher-generated prefixes when the teacher's conditional probabilities are accessible. This changes the prefix distribution while retaining the candidate construction and distillation divergence. The training experiments in this paper evaluate the interface using student-generated on-policy prefixes.

H.7 Broader Connections and Possible Extensions

The following connections suggest extensions in supervision design, interactive training, and evaluation.

Input structure and supervision quality. SepSeq studies long numerical sequences and uses separator tokens to organize attention without training [67]. This is related to our numerical and structured-text examples in the sense that the representation of local units can affect model behavior, but it changes the input rather than constructing a cross-tokenizer distillation space. Robust Preference Optimization via Dynamic Target Margins adapts preference-learning margins to improve robustness to noisy preference data [68]. It addresses the quality and strength of supervision; SimCT addresses whether teacher and student scores refer to comparable units. Together, these approaches motivate studying adaptive weighting over aligned supervision units.

Agent training data and interaction. LIMI studies learning agent behavior from a small set of curated demonstrations [69]; daVinci-Dev studies agent-native trajectories grounded in real tool interactions [70]; and daVinci-Env provides executable environments for software-engineering agent training [71]. These works motivate examining cross-tokenizer supervision on tool calls, observations, and code edits, where surface boundaries may differ between models. ARGO uses asynchronous human guidance and action-level supervision control for long-horizon research-agent training [72]. Its intervention and masking mechanisms address trajectory quality rather than tokenizer compatibility. These interaction structures suggest aligned-unit supervision over tool calls, observations, and code edits.

Evaluation beyond single responses. AgencyBench evaluates broad autonomous-agent capabilities in long contexts [73]; LongCLI-Bench evaluates extended command-line programming workflows [74]; and InnovatorBench evaluates agents conducting innovative AI research [75]. These benchmarks offer settings for evaluating whether local cross-tokenizer supervision supports long-horizon planning and execution through environment-aware rollouts and end-to-end task metrics.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope. They summarize the two key challenges in cross-tokenizer OPD, describe SimCT as a shared supervision-space construction, and keep the empirical claims limited to the evaluated models, tasks, and benchmarks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated limitations in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper states the assumptions used in the theoretical analysis and provides formal proofs for all theoretical results in Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides the experimental details necessary to reproduce the main results of SimCT. We specify the teacher and student model checkpoints, tokenizer pairs, benchmark datasets, cold-start SFT data construction, OPD training protocol, optimization hyperparameters, decoding settings, evaluation metrics, and baseline configurations. Additional implementation details, including computational setup and evaluation procedures, are provided in the appendix to support independent reproduction of the reported results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a public code release at https://github.com/sunjie279/SimCT- for reproducing the main experimental results, including the implementation of SimCT, training scripts, evaluation scripts, and baseline configurations. Because the training corpus is constructed from several publicly available datasets governed by different licenses, we do not redistribute all processed examples as a single standalone dataset. Instead, we provide scripts and instructions to download the original public datasets, apply the same filtering and preprocessing pipeline, and regenerate the training data used in our experiments. The supplemental material further specifies the required environment, data preparation procedure, and commands for running training and evaluation.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides the training and evaluation details necessary to understand the experimental results of SimCT. We specify the teacher-student model pairs, tokenizer settings, cold-start SFT data construction, OPD training protocol, benchmark datasets, evaluation metrics, decoding settings, optimizer, learning rate, batch size, training budget, and baseline configurations. The core paper includes the key settings needed to interpret the comparisons, and the appendix provides additional implementation and hyperparameter details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 1 reports means and standard deviations over 5 independent training runs for SFT and OPD, evaluating each checkpoint once; Base models use 5 decoding seeds. Figure 4 reports standard deviations over 5 independent ablation training runs. These error bars are standard deviations, not standard errors or significance tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the computational resources in Appendix F. We specify the GPU type, GPU memory, number of GPUs, approximate runtime for SFT, cross-tokenizer OPD training, and evaluation, as well as the estimated total compute cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

Justification: The research conducted in this paper conforms to the NeurIPS Code of Ethics. The work proposes a general algorithmic method for cross-tokenizer on-policy distillation and does not involve human subjects, private or sensitive personal data, deception, surveillance, or deployment in high-stakes decision-making settings. The experiments are conducted using publicly available models and datasets under their respective access terms.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses broader impacts of the proposed cross-tokenizer OPD method in Appendix A.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models or newly scraped datasets that require additional safeguards. We provide code, training and evaluation scripts, and data construction pipelines for reproducibility. Since the training data is constructed from existing publicly available datasets with their own access terms, we provide instructions for obtaining the original sources and regenerating the data rather than redistributing a new scraped dataset. Therefore, additional safeguards for controlled release of high-risk models or datasets are not applicable.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We reuse existing code, datasets, and teacher and student models, including KDFlow, public math and code datasets, 'gemma-2-2b-it' and 'Phi-4-mini-instruct' as the student models. We properly cite the original papers, repositories, and dataset/model cards, report the versions or access dates when applicable, and document the corresponding licenses and terms of use in Appendix B.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper introduces a newly generated cold-start dataset for supervised fine-tuning. The dataset is constructed by sampling and filtering prompts from existing public math and code datasets, generating candidate responses with a teacher model, and selecting responses according to task-specific correctness and quality criteria. We document the source datasets, teacher model, generation parameters, filtering and selection procedures, data format, intended use, and license information in Appendix B. To respect the licenses and redistribution terms of the original assets, we release construction scripts, source identifiers, filtering rules, metadata, and a small sample; any redistributed examples preserve source attribution and license metadata.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing, user studies, interviews, surveys, or any other research with human subjects. Therefore, participant instructions, screenshots, and compensation details are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing, user studies, interviews, surveys, or any other research with human subjects. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

Justification: LLMs were used only for manuscript writing assistance and data processing. All ideas were proposed by humans, and all generated content was carefully verified by the human authors.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.