On-Policy Distillation Across Model Families

From the basic idea to GOLD and early training instability

Fengnan Deng · September 2026

The idea in one sentence. Let the student write, let the teacher inspect the student's prefixes, and train the student with the teacher's next-token probabilities.

1 Learn where the student actually goes

A teacher can provide more than a finished answer. At each position, its probability distribution tells us which continuations it prefers and how strongly. Logit distillation uses this richer signal to train a student; in practice, the loss usually compares probabilities obtained from the logits.

The key distinction is *who supplies the continuation*. In off-policy distillation, training uses externally supplied answers, such as a fixed collection of teacher responses. In on-policy distillation (OPD), the current student generates the answer. The teacher then evaluates the prefixes that the student actually visited, including its mistakes [1]. Both approaches are useful: teacher answers provide clean examples, while student answers expose the contexts the student must learn to handle at inference time.

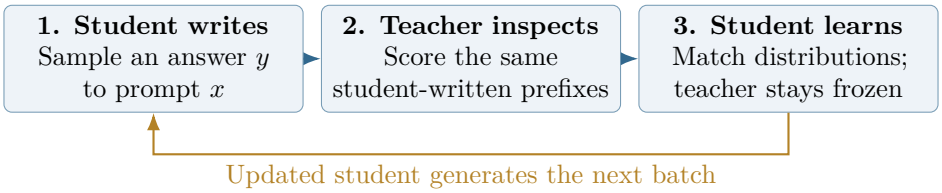

Figure 1: The OPD loop. The teacher scores the student's answer rather than replacing it with its own. With a causal mask, all positions of a completed rollout can be scored in one teacher forward pass; separate calls per token are unnecessary.

For a shared tokenizer, write $h_t = (x, y_{<t})$, and let p_T and p_θ be the teacher and student next-token distributions. A simple objective is

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x,\, y\sim p_{\theta_{\mathrm{old}}}(\cdot|x)}\left[\frac{1}{|y|}\sum_{t=1}^{|y|} D(p_T(\cdot \mid h_t), p_\theta(\cdot \mid h_t))\right]. \tag{1}$$

Here θ_{old} denotes the student used to collect the batch; sampled text is held fixed during the update. **On-policy specifies where the prefixes come from, not the direction of KL.** At each student-generated prefix h, either choice is valid [1]:

$$D_{\mathrm{forward}}(p_T, p_\theta) = \sum_v p_T(v \mid h) \log \frac{p_T(v \mid h)}{p_\theta(v \mid h)},$$

$$D_{\mathrm{reverse}}(p_T, p_\theta) = \sum_v p_\theta(v \mid h) \log \frac{p_\theta(v \mid h)}{p_T(v \mid h)}.$$

The outer expectation in (1) selects student-written *prefixes*; the inner divergence compares *all next-token alternatives* at each prefix. With full logits, either sum can be evaluated directly. If instead we sample only $v \sim p_\theta(\cdot \mid h)$, the log ratio $\log[p_\theta(v \mid h)/p_T(v \mid h)]$ estimates reverse KL. Sampling from an older student requires correction to estimate the current student's reverse KL.

Why probabilities matter. SFT supplies a selected target token; distillation also communicates the teacher's alternatives. Comparing those alternatives requires matching their meanings across the two vocabularies, which becomes difficult across model families.

2 GOLD: align the text before comparing the models

A token ID has no universal meaning. Two tokenizers can assign different IDs to the same text, use different vocabulary sizes, and split the same string at different places. Simply subtracting their logit vectors is therefore meaningless.

GOLD extends on-policy logit distillation to this setting [2]. The original blog calls it *General On-Policy Logit Distillation*; TRL calls its trainer *General Online Logit Distillation* [4]. There are two separate alignment problems.

2.1 First, align positions along the answer

Decode the student completion and tokenize that same text for the teacher, using each model's own prompt template. Locate completion tokens in a shared UTF-8 byte coordinate system. Advance the side whose token ends earlier, and close a group when both sides reach the same boundary [5].

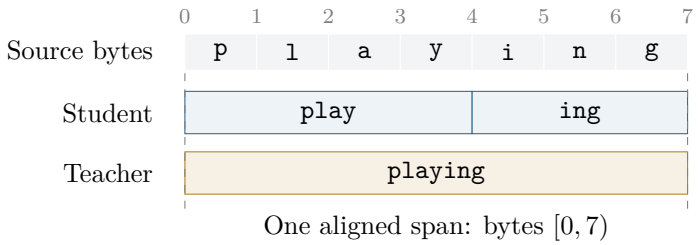

Figure 2: Toy tokenizations of the same text. The internal boundary at byte 4 is not shared, so the two student tokens form one group against the teacher token. These are illustrative splits, not outputs from a particular tokenizer.

For the observed student span, the chain rule gives

$$P_S(\texttt{playing} \mid h) = P_S(\texttt{play} \mid h)\, P_S(\texttt{ing} \mid h, \texttt{play}).$$

This is the probability of this particular token path. It does *not* construct a complete distribution over every possible next word. GOLD's group-merging rule is a practical comparison of aligned probability vectors, not an exhaustive marginalization over alternative tokenizations. Special tokens and answer boundaries also need explicit handling.

2.2 Then, align vocabulary entries

For shared vocabulary content, the hybrid GOLD loss compares mapped entries directly. For unmatched entries, ULD sorts the probabilities and pads the shorter vector, comparing their shapes without claiming that equal ranks have equal meanings [3]. In compact notation,

$$\mathcal{L}_{\text{hybrid}} = w_m \mathcal{L}_{\text{matched}} + w_u \left\| \text{pad}(\text{sort}(u_S)) - \text{pad}(\text{sort}(u_T)) \right\|_1.$$

Here u_S, u_T are the unmatched parts after sequence alignment. TRL's hybrid path uses generalized JSD on matched entries and **L1, not L2**, on the sorted unmatched entries [5]. Hybrid matching is configurable; it is not synonymous with every GOLD configuration. Byte offsets solve *where* to compare along the answer; vocabulary mapping addresses *what* is being compared.

3 My current finding: format can break before it recovers

In my cross-model GOLD runs, the student's response format can deteriorate at the beginning of training, and rollouts can become unstable. Later, the student can relearn the teacher's format. This is a practical observation from my current experiments, not a claim that every student–teacher pair follows the same trajectory.

My working explanation is a large gap between the teacher's and student's next-token distributions. Output format is itself a sequence of token decisions. If the models strongly disagree near a structural boundary, an early update can disturb the student's existing behavior before the teacher's preferred behavior has become reliable. Once a boundary is missed, subsequent generation conditions on that altered prefix, so the disruption can propagate through the answer.

This remains a hypothesis: template or stop-token mismatches, alignment errors, and update size can also contribute. Measuring the distribution gap across tokenizers itself requires meaningful alignment.

3.1 The recovery rule I use

My practical response is to retry a bad rollout. If generation keeps failing, I fall back to supervised fine-tuning on a teacher-generated answer, using a segmented loss. Subsequent student rollouts can then return to GOLD training.

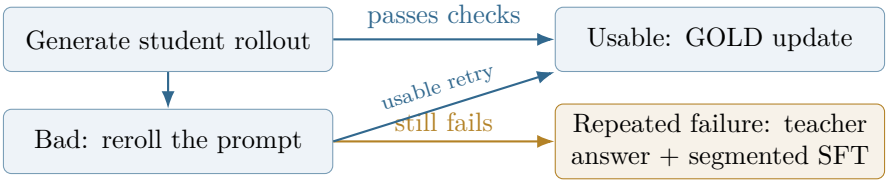

Figure 3: My recovery workflow. After either update, the next batch starts with student generation again. This is a procedural illustration; no measured recovery curve is implied.

For nonempty answer segments I_k, segmented SFT can be written as

$$\mathcal{L}_{\text{seg}} = \sum_k \alpha_k \left[-\frac{1}{|I_k|} \sum_{t \in I_k} \log p_\theta(y_t^T \mid x, y_{<t}^T) \right], \qquad \alpha_k \geq 0, \quad \sum_k \alpha_k = 1.$$

The teacher text is tokenized for the student. Segments can separate reasoning and the final answer; their exact definitions and weights are left unspecified here.

This mixes filtered on-policy rollouts with off-policy SFT: rerolling changes which trajectories reach the loss. I would track first-attempt format validity, retry success, fallback frequency, and task accuracy. Accepted rollouts alone can hide continuing instability.

Takeaway. GOLD makes cross-tokenizer teaching possible, but alignment alone does not guarantee a smooth start. In my runs, rerolling and segmented teacher-answer SFT provide a practical bridge while the student relearns the format.

References

[1] Agarwal et al. *On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes*. ICLR 2024.
[2] Patiño et al. *Unlocking On-Policy Distillation for Any Model Family*. Hugging Face, 2025; updated 2026.
[3] Boizard et al. *Towards Cross-Tokenizer Distillation: the Universal Logit Distillation Loss for LLMs*. 2024.
[4] Hugging Face. *GOLD Trainer documentation*. Accessed September 15, 2026.
[5] Hugging Face TRL. `gold_trainer.py`, alignment and hybrid-loss implementation. Accessed September 15, 2026.