REI: LEARNING IDEATION FROM REPOSITORY EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As AI systems become increasingly capable of executing and evaluating experiments, ideation is emerging as a bottleneck in automated scientific and engineering iteration. In many real-world domains, useful ideas are sparse and their validation is costly, making direct online exploration inefficient. We observe that repository evolution provides a scalable source of reusable technical experience: code changes record concrete technical hypotheses, while subsequent revisions, adoption, and reuse provide signals about their utility. Based on this observation, we propose **REI**, a Repository Evolution-driven Ideation framework that learns reusable ideation policies from repository evolution, injecting domain-specific technical experience into model parameters and reducing reliance on costly trial-and-error. We further identify a hidden failure mode in quality-driven ideation optimization: **semantic diversity collapse**. Although generated ideas can remain lexically diverse, they increasingly concentrate in a small number of dominant method families, limiting cross-task generalization and long-tail exploration. To address this problem, we introduce **S**emantic **P**rior **O**ptimization and **D**ebiasing (**SPOD**), a diversity-preserving alignment mechanism that makes method identity explicit, corrects category-level reward bias, and limits redundant accumulation within the same method family while retaining strong representatives. Experiments on public benchmarks and a real-world industrial setting demonstrate improvements in ideation quality, method diversity, cross-task generalization, and long-tail exploration.

1 INTRODUCTION

Recursive self-improvement (RSI) is becoming increasingly practical for modern AI systems (Duan et al., 2026; Zhang et al., 2026a; Huang et al., 2026). In a typical RSI loop, a system proposes a modification, executes the corresponding experiment, and retains the change if it yields measurable improvements (Novikov et al., 2025; Romera-Paredes et al., 2023; Lange et al., 2026). As implementation, training, and evaluation become increasingly automated, the primary bottleneck is shifting from *execution* to *ideation*: the central challenge is no longer merely whether a system can run experiments, but whether it can consistently identify high-value directions worth exploring. This challenge is particularly pronounced in high-frequency model and system iteration, where existing workflows increasingly rely on general-purpose language models to propose candidate improvements (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Yang et al., 2025b; Team, 2026). Such models, however, provide only weak domain-specific priors. In domains such as recommender systems, the value of an idea often becomes clear only after implementation, training, and evaluation, making poorly grounded exploration costly (Kim et al., 2026; Wang et al., 2026; Wu et al., 2026). Moreover, even when useful directions are discovered through extensive online search, the resulting experience is often tied to a particular task and search trajectory rather than distilled into reusable ideation capability that transfers across tasks.

Repository evolution provides a scalable source of the missing technical experience. A repository records a sequence of concrete engineering hypotheses: each committed modification reflects a specific design choice, while its subsequent acceptance, revision, reuse, or downstream effect provides implicit supervision about the usefulness of that direction. More importantly, repository histories naturally span heterogeneous tasks, system states, and evolution trajectories, exposing recurring patterns of what kinds of modifications tend to succeed under different conditions. Such histo-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ries therefore contain not only implementation traces, but also reusable priors over technical problem solving. Compared with learning solely through task-local online exploration, repository-scale evolution offers a way to transform accumulated engineering experience into transferable ideation knowledge (Ouyang et al., 2026; Zhang et al., 2026b), providing a substantially stronger starting point for iterative self-improvement.

However, learning from historical success alone does not ensure broad long-term exploration. We find that ideation policies can gradually become concentrated in a small number of dominant method families, improving same-task reward while weakening cross-task transfer and long-tail coverage (see Fig. 2). This stems from a self-reinforcing dynamic: once a family is favored, optimization repeatedly doubles down on similar variants, progressively crowding out underrepresented yet viable alternatives. Importantly, this collapse is often hidden at the surface level: ideas may remain lexically diverse while becoming increasingly homogeneous in their underlying methodology (see Fig. 3). Reinforcement learning can amplify this effect, and stronger verifiers do not fundamentally solve it; they can instead push the policy further toward already favored method families (see Fig. 4). A practical ideation framework therefore needs not only strong technical priors, but also an optimization mechanism that preserves semantic breadth and long-tail exploration throughout learning.

To harness repository histories for transferable ideation, we propose **REI**, a **R**epository **E**volution-driven **I**deation framework that distills historical engineering trajectories into reusable technical priors (Fig. 1). To address the method-level collapse identified above, REI introduces **SPOD** (**S**emantic **P**rior **O**ptimization and **D**ebiasing) as its core alignment mechanism. SPOD makes method-level semantic structure explicit, corrects category-level reward imbalance, and employs top-1 retention to suppress redundant variants within a dominant family while preserving its strongest representative, thereby preserving broader exploration without discarding high-quality directions. We evaluate REI on public benchmarks and a real-world industrial recommender system. Across these settings, REI consistently improves ideation quality, semantic method diversity, cross-task generalization, and long-tail exploration. In end-to-end deployment on the industrial recommender system, REI outperforms frontier general-purpose models and human experts, while delivering sustained gains under continual iteration.

Our main contributions are as follows:

- **A repository-evolution framework for ideation.** We propose REI, which learns reusable ideation policies from repository evolution for continual model and system improvement.
- **A diagnosis of hidden method-level collapse.** We reveal a hidden form of method-level collapse in quality-driven ideation optimization: method diversity can collapse even when token entropy and lexical diversity remain relatively stable, and stronger verifiers can further exacerbate this effect.
- **SPOD for diversity-preserving alignment.** We propose SPOD, which uses method-level semantic annotation, category-aware reward correction, and top-1 retention to mitigate method-level collapse during policy optimization.
- **Validation on public and industrial domains.** We evaluate REI and SPOD on public benchmarks and a real-world industrial recommender system, showing consistent gains in ideation quality, diversity, cross-task generalization, and long-tail exploration. In the industrial setting, they further yield sustained gains over continual iteration, including a +4.86% GMV lift and a +1.82% advertising revenue lift.

2 Related Work

2.1 Agentic Automation of Engineering Iteration

AI is increasingly used to automate engineering iteration, spanning code generation, experiment execution, literature-informed proposal generation, and closed-loop optimization (Guo et al., 2024; Hong et al., 2024; Trirat et al., 2025; Nam et al., 2025; Hollmann et al., 2023; Tang et al., 2025). More recent systems further incorporate the resulting interaction trajectories into reinforcement learning, with the goal of turning online trial-and-error into improved ideation capability. In recommender systems, for example, agentic pipelines have been developed to iteratively refine models

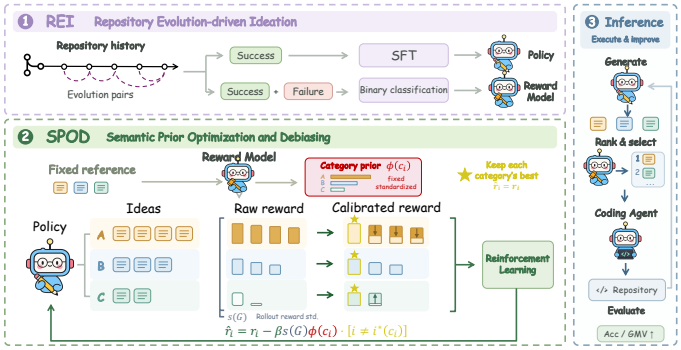

Figure 1: REI learns reusable ideation policies from repository evolution, while SPOD prevents semantic collapse by debiasing method families and retaining strong representatives.

and strategies through repeated interaction with online feedback (Zhang et al., 2026c). Despite these advances, most existing approaches still rely on a general-purpose LLM as the initial ideation policy, leaving domain-specific technical knowledge to be acquired through costly task-local exploration. REI instead learns this knowledge from repository evolution, turning accumulated engineering trajectories into reusable priors before online iteration begins.

2.2 Diversity and Exploration in RL Alignment

A related line of work studies diversity preservation during RL alignment, often building on group-relative policy optimization (Shao et al., 2024) and introducing exploration controls at the token or policy-update level. Entropy regularization explicitly adds an entropy bonus to discourage premature policy concentration (Schulman et al., 2017), while entropy-based advantage adjustment assigns stronger learning signals to high-entropy tokens (Cheng et al., 2025). Other approaches modify the policy update rule itself. CISPO clips only the importance-sampling weight while retaining the token-level policy gradient (Chen et al., 2025); DAPO relaxes the clipping constraint to allow initially low-probability positive tokens to receive larger updates (Yu et al., 2025); and STEER reweights updates according to their predicted effect on entropy, suppressing updates that would induce excessive entropy reduction (Hao et al., 2026). These methods primarily preserve exploration in the token-level policy distribution. In contrast, SPOD focuses on semantic diversity at the method-family level: it explicitly represents method identity and corrects category-level reward imbalance to reduce excessive concentration on a small set of dominant solution families.

3 Method

3.1 Learning Ideation Experience from Repository Histories

Rather than acquiring ideation ability primarily through costly online trial-and-error, we learn an ideation policy directly from repository evolution. Repository histories provide scalable supervision by recording concrete modifications together with their subsequent empirical outcomes, allowing accumulated engineering experience to be converted into training signals for ideation.

For each target system, we extract training trajectories from its repository history, each comprising a scenario context, a proposed modification, and its observed outcome. We train the ideation policy in two stages. First, we perform supervised fine-tuning on successful historical modifications, grounding the policy in domain-specific technical patterns. Second, we train a verifier on both successful and failed historical outcomes to predict modification utility, and use it to refine the policy via reinforcement learning, optimizing toward directions with stronger empirical gains. During downstream inference, the policy samples candidate ideas that are ranked by the verifier; the top proposals are then dispatched to a coding agent for automated implementation (Fig. 1).

3.2 Analysis: Diversity Collapse in Idea Generation

Learning from repository histories provides the ideation policy with a stronger starting point, yet subsequent alignment still determines how exploration is distributed across method families. During iterative optimization, some families are easier to reinforce than others, potentially narrowing the range of solution directions explored by the policy.

This effect is particularly relevant to recommender systems. For example, modifying the loss formulation can quickly improve the current offline metric and is therefore readily rewarded by a verifier, whereas introducing multi-interest user modeling (Zhou et al., 2017; Lv et al., 2026) may yield smaller initial gains despite its potential value over continued iteration. Repeatedly favoring directions with stronger short-term rewards can therefore concentrate exploration on a narrow subset of method families.

Empirical analysis reveals a systematic collapse in method-level diversity during reinforcement learning. Although the initial policy already prefers certain solution families, standard reinforcement learning amplifies this tendency and concentrates generation on a small number of dominant families. Appendix C gives the protocol and metric definitions for the three analyses below.

Long-tail and cross-task performance decline. As training progresses, performance on the training task continues to improve. On the same task, GRPO increases both the mean verifier reward and the best-of-8 reward, as shown by the solid lines in Figure 2(a). However, these gains do not transfer to the out-of-distribution cross-task setting, where the mean reward slightly decreases and the best-of-8 reward changes little (dashed lines in Figure 2(a)). The discrepancy is more pronounced across method families. When method families are grouped by their share among generated candidates and Hit@1 is measured within each group, GRPO improves the head family while degrading the mid- and tail-frequency families (Figure 2(b)). Thus, improvements on the training objective can coincide with weaker cross-task generalization and reduced coverage of long-tail method families.

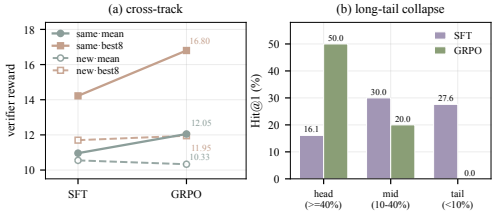

Figure 2: Diversity collapse after RL alignment. **(a)** Same-task verifier reward improves under GRPO, but cross-task gains disappear. **(b)** Method families are grouped by their generated share: head ($\geq$ 40%), mid (10–40%), and tail ($<$ 10%). GRPO improves the head family while degrading the mid and tail families.

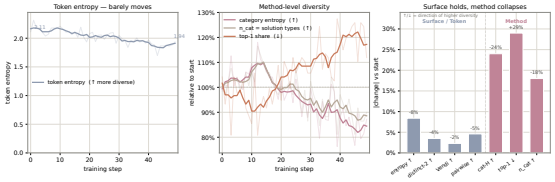

Figure 3: Surface diversity holds while method diversity collapses. *Left*: token entropy over training (2.11 $\rightarrow$ 1.94; axis starts at zero). *Middle*: method-level trajectories, normalized to step 0. *Right*: magnitude of change at final checkpoint relative to SFT initialization; arrows mark higher diversity.

Lexical diversity does not explain the collapse. The degradation in cross-task and long-tail performance suggests that exploration becomes increasingly concentrated during training. However, this concentration cannot be explained by a corresponding collapse in token- or lexical-level diversity. Token entropy exhibits only a marginal decline throughout training (Figure 3, left), while lexical dissimilarity changes by less than 5% relative to the SFT initialization (Figure 3, right).

Instead, the collapse is substantially more pronounced at the level of method semantics. An offline LLM maps each generated idea into a method taxonomy $\mathcal{C}$. During training, category entropy and the number of distinct method families n_{cat} decrease steadily, while the share of the most frequent method family increases monotonically (Figure 3, middle). At the final checkpoint, lexical statistics change by only 2–8%, whereas method-level statistics change by 18–29%. The policy therefore remains relatively diverse in surface realization while becoming increasingly concentrated in its underlying method families.

Stronger optimization further amplifies the collapse. Another possible explanation is that the observed concentration is primarily caused by exposure imbalance in the training data or verifier noise. The experimental results do not support this interpretation. Rebalancing the training samples across method categories leaves the collapse largely unchanged. More importantly, improving verifier fidelity, measured as pairwise ranking accuracy between verifier scores and realized gains, does not mitigate the effect; instead, it further increases method-level concentration. As shown in Figure 4, the head-method share rises monotonically from 49.1% to 58.8% to 62.5% as verifier fidelity increases.

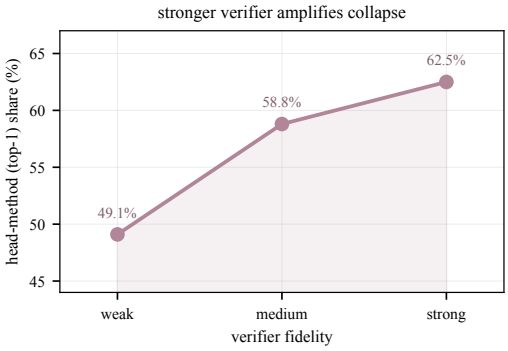

Figure 4: A higher-fidelity verifier accelerates the collapse: the head-method (top-1) share rises monotonically across the weak/medium/strong verifiers (fidelity 0.55/0.67/0.83).

Under group-relative updates such as GRPO, category-level reward differences can become self-reinforcing. Candidates from method families that consistently receive higher rewards are more likely to obtain positive advantages, causing subsequent policy updates to further favor these families. Higher-fidelity verification may amplify this effect by preserving the current reward ordering more faithfully. Taken together, these results indicate that the observed collapse occurs primarily at the level of method families rather than surface language. Preserving token-level or lexical diversity is therefore insufficient for maintaining broad exploration. Effective ideation alignment should explicitly account for method-level semantics and reduce systematic reward advantages that repeatedly concentrate optimization on already favored method families.

3.3 SPOD: Semantic Prior Optimization for Diversity-Preserving Ideation

To counter method-level concentration, SPOD introduces a three-step alignment pipeline (Fig. 1). For each proposal group, it first assigns method categories through semantic annotation, then computes reward corrections from category reward priors, and finally applies them while exempting each category's highest-reward candidate.

Method-Level Semantic Annotation. Because the observed collapse occurs at the level of method families rather than surface wording, SPOD explicitly represents method identity. For each candidate idea y, we assign a method category $c \in \mathcal{C}$ and prepend the corresponding tag, yielding $\tilde{y} = [\mathrm{TAG}_c] \circ y$, where $\circ$ denotes concatenation. This representation allows subsequent optimization to distinguish candidates by their underlying method family rather than only their lexical form. When a ground-truth taxonomy is available, it is used directly; otherwise, the category is assigned by an offline LLM annotator.

Category-Aware Reward Correction. Under the same objective, some method categories consistently receive higher rewards than others. Without correction, these category-level reward differences can cause group-relative optimization to repeatedly favor the same method families. SPOD compensates for this effect by adjusting candidate rewards according to the reward prior of their corresponding method category.

A fixed reference set $\mathcal{D}_{\mathrm{ref}}$ of categorized ideas is used to estimate the category-level reward prior. Let $R_i = R(x, y_i)$ denote the raw reward of candidate y_i with category label c_i. For a sampled candidate group $G = \{y_1, \ldots, y_K\}$ with reward standard deviation $s(G)$, the correction is

$$\mu_c = \mathbb{E}_{(x,y)\in\mathcal{D}_{\mathrm{ref}},\, \mathrm{cat}(y)=c}\left[R(x,y)\right], \qquad \varphi(c) = \frac{\mu_c - \bar{\mu}}{\sigma_\mu}, \qquad \Delta_i = \beta\, s(G)\, \varphi(c_i), \tag{1}$$

where $\bar{\mu}$ and σ_μ denote the mean and standard deviation of the category-wise average rewards, respectively, and $\beta > 0$ controls the correction strength. A larger $\varphi(c)$ corresponds to a stronger positive reward prior for category c.

The corrected reward is

$$\tilde{R}_i = R_i - \Delta_i. \tag{2}$$

Candidates from categories with stronger positive reward priors receive larger downward corrections, thereby leveling the playing field across method families.

Inspired by debiasing techniques in recommender systems (Chen et al., 2020; Gao et al., 2023; 2022; Lin et al., 2024), this mechanism decouples category-level bias from individual candidate quality, thereby neutralizing systematic family advantages while keeping candidate-level quality competition intact. The following theorem formalizes this effect.

Theorem 1. *In the calibrated reward formulation $r'_i = r_i - \beta \cdot s(G) \cdot \varphi(c_i)$, the normalized fixed-dispersion score $\widetilde{A}_i$ satisfies*

$$\widetilde{A}_i = \underbrace{\frac{\delta q_i - \overline{\delta q}_G}{s(G)}}_{A_i^{\text{intra}}} + \left(1 - \frac{\beta \cdot s(G)}{\sigma_\mu}\right) \underbrace{\frac{\mu_{c_i} - \bar{\mu}_G}{s(G)}}_{A_i^{\text{inter}}}, \tag{3}$$

where A_i^{intra} is the category-centered quality term and A_i^{inter} is the category-mean reward term. The correction leaves the former unchanged and linearly contracts the latter.

The theorem shows that category-aware reward correction preserves the category-centered quality term while contracting the category-mean reward term. It therefore reduces the component associated with systematic reward differences across method families without weakening candidate-level quality competition. The full statement and proof are provided in Appendix A.

Top-1 Retention. Applying the same correction to every candidate within a method category would also suppress its strongest representative together with weaker redundant variants. SPOD therefore exempts the top candidate within each category and applies the correction only to the remaining candidates:

$$i^\star(c) = \arg\max_{j \in \mathcal{G}_c} R(x, y_j), \qquad \tilde{R}_i = R_i - \Delta_i \cdot \mathbb{I}(i \neq i^\star(c_i)), \tag{4}$$

where $\mathcal{G}_c$ denotes the set of candidates in group G that belong to category c. The correction is thus concentrated on redundant variants within a method family while preserving its strongest representative, reducing repeated reinforcement of locally similar candidates without removing strong candidates from cross-category competition.

Taken together, SPOD transforms standard reward optimization into a diversity-preserving search: semantic tags expose the underlying method structure, reward correction contracts inter-family bias, and top-1 retention shields the best local candidates from over-penalization. By jointly addressing method-level collapse without discarding high-quality solutions, SPOD maintains a broad and healthy exploration frontier throughout alignment.

4 Experiment

In this section, we answer three questions: (1) Does REI, coupled with SPOD, deliver superior real-world performance over frontier LLMs and sustain multi-round iteration? (Section 4.2, Section 4.3) (2) Can SPOD prevent method collapse and maintain broad exploration across unseen tasks and long-tail domains? (Section 4.4, Section 4.5) (3) How does each design component in SPOD affect the final quality-diversity profile? (Section 4.6)

4.1 Experimental Setup

Our experiments combine online and offline evaluations in an industrial AI system, offline evaluation on a public benchmark built from nanoGPT speedrun leaderboard records, and additional offline evaluation on rustc-perf (Appendix G). Online evaluation uses a production recommender system, where generated ideas undergo the full implementation, training, validation, and deployment pipeline. Offline evaluations use ground-truth method categories for the public benchmarks and LLM-derived reference categories for the industrial system.

Evaluation settings. We use Qoder as the coding agent to implement model-generated ideas. Industrial experiments use Qwen3-8B (Yang et al., 2025a); public experiments use Qwen2.5-0.5B (Yang et al., 2024), with a second public benchmark (rustc-perf, Appendix G) using Qwen3-1.7B. For each test case, the model generates eight ideas, ranked by the reward model.

Table 1: End-to-end industrial validation. Success@k is judged by true execution after implementation and training. Higher is better for all metrics.

Metric	Human	Claude 4.8-Opus	DeepSeek V4 Pro	GPT 5.5	REI 8B (base)	REI 8B (+SPOD)
Success@1	0.32	0.26	0.22	0.31	0.68	**0.69**
Success@3	0.49	0.43	0.29	0.46	0.79	**0.83**

Metrics and method categories. For online evaluation, we report Success@k, the fraction of prompts with at least one loss-reducing idea among the top-k candidates. For offline evaluation, we assess category alignment (Hit@1, Hit@8, Coverage, TV to Reference Prior) and diversity (Distinct Method Count, Top-Family Share, Max Within-Family Count, Single-Family Collapse Rate). Public categories are ground-truth labels; industrial categories are LLM-extracted from internal repositories. Full metric definitions are in Appendix B.

Evaluation slices and baselines. We report aggregate results, cross-task transfer, and long-tail performance after excluding the dominant category. We compare SPOD with SFT, GRPO (Shao et al., 2024), and diversity-oriented baselines, including entropy regularization (Schulman et al., 2017), entropy-based advantage adjustment (Cheng et al., 2025), CISPO (Chen et al., 2025), STEER (Hao et al., 2026), and DAPO (Yu et al., 2025). Industrial experiments additionally compare REI with and without SPOD to isolate SPOD's effect under real execution. Baseline details, benchmark construction, implementation, data splits, and extended results appear in Appendix D and Appendix B.

4.2 End-to-End Industrial Validation

To assess whether REI improves system performance under actual execution, model-generated proposals are first implemented by a coding agent and then evaluated through training and execution in the industrial environment. As shown in Table 1, REI achieves substantially higher Success@1 and Success@3 than both human experts and general-purpose LLMs. Specifically, REI-8B reaches 0.68 Success@1 and 0.79 Success@3, compared with 0.32/0.49 for human experts and 0.31/0.46 for GPT-5.5. Incorporating SPOD further improves these metrics to 0.69 and 0.83, respectively. The larger improvement on Success@3 is consistent with the design objective of SPOD, which aims to preserve broader method coverage across multiple candidates while reducing redundant variants within the same method family.

4.3 Multi-round Iteration

Real-world optimization typically unfolds over repeated iterations rather than a single round of improvement. To evaluate performance under continual iteration, we conduct experiments in both a live production loop and an offline replay setting based on public speedrun data. In the production loop, starting from the deployed baseline, REI drives 20 consecutive iterations over two weeks and yields a cumulative **+23.6%** improvement in Hitrate@200 relative to the initial baseline. Adding SPOD on top of REI brings a further **+4.2%** improvement in Hitrate@200. In the final online A/B test, live traffic is randomly split between treatment and control over a fixed observation window. The resulting system further delivers a **+4.86%** lift in GMV and a **+1.82%** lift in advertising revenue.

For the public speedrun data, an offline replay simulation is conducted by sorting historically accepted improvements within each method family according to their realized gains. At each round, a method family is sampled from the trained model's measured output distribution, and the next unused realized gain from that family is assigned. As shown in Figure 5, GRPO increasingly concentrates on a dominant method family and slows later in the horizon, whereas SPOD maintains broader coverage across method families and continues to accumulate reward. At $T = 30$, SPOD achieves a higher cumulative reward than GRPO (246.5 vs. 197.2), and the advantage persists under cross-task transfer (218.1 vs. 171.6; see Appendix H). These replay results further indicate that preserving method-family coverage is beneficial for sustained multi-round optimization.

4.4 Overall Quality–Diversity Performance

As shown in Table 2, existing baselines improve only part of the quality profile and do not consistently balance top-ranked accuracy, multi-candidate coverage, and distributional alignment. SPOD achieves the strongest results on all four metrics on the nanoGPT benchmark and shows the same overall trend in the industrial offline evaluation, indicating that SPOD improves category-level candidate quality while bringing the generated method distribution closer to the reference distribution.

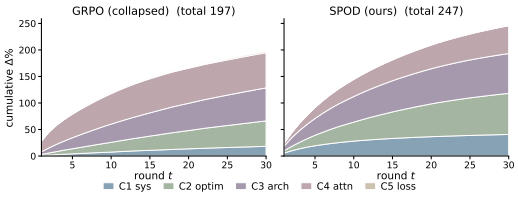

Figure 5: Replay at T=30: colored bands show cumulative reward by method family (C1–C5; Appendix B). SPOD maintains broader family coverage and achieves higher cumulative reward.

The diversity results show a similar pattern. As shown in Table 3, SPOD achieves a higher Distinct Method Count and lower Top-Family Share and Max Within-Family Count than GRPO and the other post-training baselines, while also maintaining a low Single-Family Collapse Rate. These results indicate that SPOD substantially mitigates semantic concentration toward a small number of dominant method families.

Table 2: Overall quality comparison on the industrial offline analysis and the nanoGPT benchmark. Higher is better for Hit@1, Hit@8, and Coverage, while lower is better for TV to Reference Prior.

Method	Industrial Offline				NanoGPT Benchmark			
	Hit@1↑	Hit@8↑	Coverage↑	TV↓	Hit@1↑	Hit@8↑	Coverage↑	TV↓
SFT	0.687	0.960	0.960	0.238	0.391	0.783	0.707	0.297
GRPO	0.698	0.949	0.949	0.232	0.217	0.696	0.565	0.498
STEER	0.730	0.944	0.944	0.189	0.217	0.739	0.641	0.475
DAPO	0.741	0.960	0.960	0.266	0.304	0.739	0.641	0.498
Ent. Reg.	0.698	0.860	0.860	0.283	0.217	0.783	0.652	0.524
CISPO	0.640	0.897	0.897	0.214	0.217	0.696	0.565	0.515
Entro. Adv.	0.701	0.870	0.870	0.171	0.261	0.826	0.707	0.395
SPOD	**0.787**	**0.970**	**0.970**	**0.153**	**0.435**	**0.913**	**0.870**	**0.123**

Table 3: Method diversity on the industrial offline analysis and nanoGPT benchmark. ↑/↓ indicate the preferred direction; bold marks the best value. SPOD is shaded.

Method	Industrial Offline				NanoGPT Benchmark			
	Distinct↑	Top-Fam↓	Max-Fam↓	Collapse↓	Distinct↑	Top-Fam↓	Max-Fam↓	Collapse↓
SFT	3.054	0.420	5.247	**0.043**	3.679	0.415	3.929	**0.000**
GRPO	2.544	0.500	5.441	0.140	3.000	0.621	5.071	0.071
STEER	2.906	0.472	4.903	0.075	3.107	0.603	4.964	0.071
DAPO	3.033	0.480	4.989	0.071	3.071	0.612	5.000	0.071
Ent. Reg.	3.269	0.428	4.681	0.084	3.000	0.643	5.286	0.071
CISPO	2.690	0.488	4.945	0.065	2.893	0.643	5.250	0.071
Entro. Adv.	2.961	0.476	4.659	0.090	3.429	0.522	4.393	0.036
SPOD	**3.321**	**0.341**	**4.021**	0.056	**3.786**	**0.335**	**3.571**	**0.000**

4.5 Cross-Task and Long-Tail Evaluation

Cross-task generalization. As shown in Table 4 (left), SPOD achieves the highest Hit@8 and Coverage, and the lowest TV to Reference Prior on the nanoGPT cross-task slice. This suggests SPOD better preserves access to relevant method families under task transfer. Although SFT achieves a higher Hit@1, it performs worse on the remaining category-alignment metrics. On the industrial cross-task slice, SPOD achieves the highest Hit@1, Hit@8, and Coverage (Appendix F, Table 6).

Long-tail exploration. After excluding the dominant head method family on nanoGPT (Table 4, right), SPOD ties for the highest Hit@1 and achieves the highest Hit@8 and Coverage, with a lower

TV than Entro. Adv. On the industrial long-tail slice, SPOD also achieves the highest Hit@1 and Hit@8, exceeding the strongest baseline for each metric by 7.3 and 6.8 percentage points, respectively. These results show that SPOD's advantages extend to non-head method families.

Table 4: Quality comparison on two challenging nanoGPT-benchmark slices: cross-task (left) and long-tail (right). Industrial offline results are reported in Appendix F.

Method	Cross-Task				Long-Tail			
	Hit@1↑	Hit@8↑	Coverage↑	TV↓	Hit@1↑	Hit@8↑	Coverage↑	TV↓
SFT	**0.500**	0.714	0.625	0.308	0.429	0.762	0.698	0.155
GRPO	0.214	0.643	0.500	0.476	0.333	0.619	0.540	0.286
STEER	0.214	0.643	0.554	0.456	0.333	0.714	0.651	0.225
DAPO	0.286	0.714	0.625	0.476	**0.476**	0.714	0.651	0.261
Ent. Reg.	0.357	0.643	0.536	0.449	0.333	0.667	0.587	0.207
CISPO	0.214	0.643	0.500	0.435	0.286	0.667	0.587	0.265
Entro. Adv.	0.214	0.786	0.696	0.393	0.286	0.762	0.698	0.096
SPOD	0.286	**0.929**	**0.857**	**0.134**	**0.476**	**0.905**	**0.857**	**0.094**

4.6 Ablation Study

Table 5 reports a staged ablation of SPOD on the nanoGPT benchmark. The order follows the construction of SPOD: category-aware reward correction requires method-level annotation, while top-1 retention further relies on reward scores within each category. Method-level annotation alone improves Hit@8 and Coverage over GRPO. Adding category-aware reward correction further improves Hit@1, TV to Reference Prior, and the diversity metrics. Finally, incorporating top-1 retention yields the strongest overall quality–diversity performance. The progressive improvements demonstrate the contribution of each component to the complete SPOD formulation.

Table 5: Component ablation of SPOD on the nanoGPT benchmark. Ann. denotes method-level annotation, RC denotes category-aware reward correction, and Top1 denotes top-1 retention.

Method	Quality				Diversity			
	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
GRPO	0.217	0.696	0.565	0.498	3.000	0.621	5.071	0.071
+ Ann.	0.217	0.826	0.707	0.431	3.000	0.643	4.607	0.036
+ Ann. + RC	0.391	0.826	0.761	0.199	3.750	0.388	3.857	**0.000**
+ Ann. + RC + Top1	**0.435**	**0.913**	**0.870**	**0.123**	**3.786**	**0.335**	**3.571**	**0.000**

4.7 Further Analysis

The appendix reports additional analyses: sensitivity to the debiasing strength β and exemption depth k (Appendix E), the cross-task and long-tail slices (Appendix F), replay simulations and case studies (Appendix H, I), diversity dynamics across the token, surface, and method layers (Appendix J), consistency across LLM-based extractors under different annotation schemes (Appendix K), multi-seed statistical tests (Appendix L) and a second public benchmark (Appendix G). The main conclusions remain unchanged across these analyses.

5 Conclusion

We presented REI, a framework for learning reusable ideation policies from repository histories. We showed that standard reward optimization can cause hidden method-level collapse, even when surface diversity remains largely intact. We therefore introduced SPOD to preserve method coverage during alignment. Experiments on public benchmarks and a real-world industrial recommender system show consistent gains in quality, diversity, and practical effectiveness.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, code debugging, and consistency checks of the manuscript and mathematical exposition. All scientific ideas, experimental designs, data construction procedures, analyses, and conclusions were developed and verified by the authors. The authors reviewed all AI-assisted outputs and take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the complete datasets, source code, configuration files, evaluation scripts, and trained model checkpoints for both public benchmarks, nanoGPT and Rust. These artifacts cover data preprocessing, model training, inference, and metric computation, allowing the reported public experiments to be reproduced end to end. Dataset construction, experimental protocols, hyperparameters, and implementation details are documented in Appendix B.

REFERENCES

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 41:1 – 39, 2020. URL https://api.semanticscholar.org/CorpusID:222177417.

MiniMax Aili Chen, Aonian Li, Bangwei Gong, Binyan Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chun-Hao Zhang, Chunhui Du, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Enwei Jiao, Haigang Zhou, Haimo Zhang, Han Ding, Haohai Sun, Hao Feng, Huaiguang Cai, Haichao Zhu, Jian Sun, Jiaqi Zhuang, Jia-Yao Cai, Jiayuan Song, Jin Zhu, Jingyang Li, Jinhao Tian, Jinli Liu, Junhao Xu, Junjie Yan, Junteng Liu, Junxia He, Kaiyi Feng, Ke Yang, Ke Xiao, Le Han, Leyang Wang, Lian-Chun Yu, Li Feng, Lin Li, Lin Zheng, Linge Du, Ling Yang, Lu Zeng, Ming-Yuan Yu, Mingliang Tao, Ming-Yuan Chi, Mozhi Zhang, Mu-Jie Lin, Nan Hu, Nongyu Di, Peng Gao, Peng-Fei Li, Peng Zhao, Qi-Bing Ren, Qidi Xu, Qile Li, Qin Wang, Rong Tian, Ruitao Leng, Shaoxiang Chen, Shaoyu Chen, Shengmin Shi, Shitong Weng, Shuchang Guan, Shu Yu, Si-Jie Li, Song-Quan Zhu, Tengfei Li, Tian-Yi Cai, Tianrun Liang, Weiyu Cheng, Weize Kong, Wenkai Li, Xiancai Chen, Xiangjun Song, Xiao Luo, Xiaoyan Su, Xiao-Bo Li, Xiaodong Han, Xi-Yong Hou, Xuan wen Lu, Xun Zou, Xuyang Shen, Yanting Gong, Yan Ma, Yang Wang, Yiqi Shi, Yi-Fan Zhong, Yonghong Duan, Yongxiang Fu, Yong Hu, Yu Gao, Yuanxiang Fan, Yufeng Yang, Yuhao Li, Yulin Hu, Yunan Huang, Yunji Li, Yun Xu, Yuxin Mao, Yuxuan Shi, Yuze Wenren, Zehan Li, Zelin Li, Z. Tian, Zhengmao Zhu, Zhe yu Fan, Zhenzhen Wu, Zhi-Chao Xu, Zhihang Yu, Zhiheng Lyu, Zhuo Jiang, Zibo Gao, Zijia Wu, Zijian Song, and Zijun Sun. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *ArXiv*, abs/2506.13585, 2025. URL https://api.semanticscholar.org/CorpusID:279402731.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. In *AAAI Conference on Artificial Intelligence*, 2025. URL https://api.semanticscholar.org/CorpusID:279410424.

Yi-Xiang Duan, Ying Liu, Zi-Rui Tang, Hao-Dong Chen, Jun Zhou, Yu-Mou Liu, Bang-Rui Xu, Yu-Kai Wu, Si-Di Chen, Yu-Han Zhou, Hao-Yu Wang, Xiao-You Yu, Shao-Kun Han, Xu-Zhou Zhu, Leocardia Zhou, Bo-Lin Lu, Wei Zhou, Jia-Cheng Liu, Nuo-Zhou Fang, Jia-Xin Tian, Ruo-Yu Chen, Yuxuan Li, Kai Zuo, Kai-Yan Zhang, Qian-Yu Yang, Zi-Jie Wang, Jian-Tao Qiu, Cong-Hui He, Guo-Liang Li, Bo-Wen Zhou, Zhi-Yuan Liu, Zhoufutu Wen, Jian Kang, Xuan-He Zhou, and Fan Wu. The last ai built by humans: Toward genuine recursive self-improvement. 2026. URL https://api.semanticscholar.org/CorpusID:291905750.

Chongming Gao, Wenqiang Lei, Jiawei Chen, Shiqi Wang, Xiangnan He, Shijun Li, Biao Li, Yuan Zhang, and Peng Jiang. Cirs: Bursting filter bubbles by counterfactual interactive recommender system. *ACM Transactions on Information Systems*, 42:1 – 27, 2022. URL https://api.semanticscholar.org/CorpusID:247940070.

Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqin Wang, Zhong Zhang, and Xiangnan He. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023. URL https://api.semanticscholar.org/CorpusID:259501995.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *ArXiv*, abs/2402.17453, 2024. URL https://api.semanticscholar.org/CorpusID:268033675.

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. Rethinking entropy interventions in RLVR: An entropy change perspective. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31105–31133, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1436. URL https://aclanthology.org/2026.acl-long.1436/.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=9WSxQZ9mG7.

Sirui Hong, Yizhang Lin, Bangbang Liu, Bin-Hao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Lingyao Zhang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wen-Yi Wang, Xiangru Tang, Xiang Lu, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zong-Ze Xu, Chenglin Wu, Li Zhang, Min Yang, and Xiawu Zheng. Data interpreter: An llm agent for data science. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:268063292.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning LLM from zero data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=96apU6YzSO.

Sein Kim, Sang Yoon Park, Hongseok Kang, Wonjoong Kim, Jimin Seo, Yeonjun In, Kanghoon Yoon, and Chanyoung Park. Self-evolverec: Self-evolving recommender systems with llm-based directional feedback. *ArXiv*, abs/2602.12612, 2026. URL https://api.semanticscholar.org/CorpusID:285606704.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lKEdGCoDNC.

Si-Yi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. How do recommendation models amplify popularity bias? an analysis from the spectral perspective. *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 2024. URL https://api.semanticscholar.org/CorpusID:269214175.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Nicolaus Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *ArXiv*, abs/2408.06292, 2024. URL https://api.semanticscholar.org/CorpusID:271854887.

Zhihao Lv, Long Zhang, Ailong He, Shuzhi Cao, Shuguang Han, and Ju-Feng Chen. Daian: Deep adaptive intent-aware network for ctr prediction in trigger-induced recommendation. *ArXiv*, abs/2602.13971, 2026. URL https://api.semanticscholar.org/CorpusID:285616489.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan O Arik, and Tomas Pfister. MLE-STAR: Machine learning engineering agent via search and targeted refinement. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vS1M06Px6u.

Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav M. Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *ArXiv*, abs/2506.13131, 2025. URL https://api.semanticscholar.org/CorpusID:278658695.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468 – 475, 2023. URL https://api.semanticscholar.org/CorpusID:266223700.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024. URL https://api.semanticscholar.org/CorpusID:267412607.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. AI-researcher: Autonomous scientific innovation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=kQWyOYUAC4.

AgentX Team. Agentx: Towards agent-driven self-iteration of industrial recommender systems. *ArXiv*, abs/2606.26859, 2026. URL https://api.semanticscholar.org/CorpusID:289627646.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. AutoML-agent: A multi-agent LLM framework for full-pipeline autoML. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=p1UBWkOvZm.

Hao-Chen Wang, Yi Wu, Daryl Chang, Li Wei, and Lukasz Heldt. Self-evolving recommendation system: End-to-end autonomous model optimization with llm agents. *ArXiv*, abs/2602.10226, 2026. URL https://api.semanticscholar.org/CorpusID:285470525.

Xidong Wu, Zhuan Yue, Ruo-Qiao Wei, Hang Chen, Di Bai, Jintao Liu, Xin-Yi Wang, Xue Wang, Luo-Shu Wang, and Xin Cheng. Agenticrectune: Multi-agent with self-evolving skillhub for recommendation system optimization. *ArXiv*, abs/2604.26969, 2026. URL https://api.semanticscholar.org/CorpusID:287915829.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Nicolaus Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *ArXiv*, abs/2504.08066, 2025. URL https://api.semanticscholar.org/CorpusID:277741107.

An Yang, An-Feng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Jingren Zhou, Junyan Lin, Kai Dang, Keqin Bao, Ke-Pei Yang, Le Yu, Li-Chun Deng, Mei Li, Min Xue, Ming-Ze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tian-Yi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Ying-Er Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhen-Ru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. 2025a. URL https://api.semanticscholar.org/CorpusID:278602855.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin 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, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhen-Ru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL https://api.semanticscholar.org/CorpusID:274859421.

Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuan-Te Li, Haoran Pan, Yuge Zhang, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. R&d-agent: An llm-agent framework towards autonomous data science. 2025b. URL https://api.semanticscholar.org/CorpusID:278782890.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=pUpzQZTvGY.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eC4ygDs02R.

Wenlin Zhang, Xiangyang Li, Qiyuan Ge, Kuicai Dong, Pengyue Jia, Xiaopeng Li, Zijian Zhang, Maolin Wang, Yichao Wang, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. Exploring recommender system evaluation: A multi-modal llm agent framework for a/b testing. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '26, pp. 2878–2889, New York, NY, USA, 2026c. Association for Computing Machinery. ISBN 9798400722585. doi: 10.1145/3770854.3785688. URL https://doi.org/10.1145/3770854.3785688.

Guorui Zhou, Cheng-Ning Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xi-Wang Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2017. URL https://api.semanticscholar.org/CorpusID:1637394.

A Theoretical Analysis

In this section, we formulate a rigorous theoretical foundation for our proposed calibration framework under Group Relative Policy Optimization (GRPO). We first formalize the problem by characterizing how category-level systemic reward bias propagates through the empirical group-normalization mechanism of GRPO (Section A.1), establishing an exact dual-channel algebraic decomposition of the policy advantage (Section A.2). Subsequently, we prove the structural necessity of coupling the penalty term with the group sample dispersion and a standardized category deviation metric (Section A.3). Building upon these results, we establish the uniform linear contraction theorem of the category-mean reward channel and provide formal arguments justifying why a soft shrinkage operator is preferable to hard zero-cancellation (Section A.4).

A.1 Problem Formulation and Additive Bias Model

Let $\mathcal{X}$ denote the prompt domain, and let $\mathcal{Y}$ represent the sequence space of generated responses. Let $\pi_\theta(y \mid x)$ denote the parameterized autoregressive policy. We consider a discrete semantic category universe $\mathcal{C} = \{1, 2, \ldots, C\}$ consisting of $C \in \mathbb{N}^+$ distinct categories. We assume the existence of a deterministic semantic classifier $c : \mathcal{Y} \to \mathcal{C}$ that maps each response y to its underlying category label $c(y) \in \mathcal{C}$.

Definition 1 (Additive Category-Level Bias Model)**.** For any response $y \in \mathcal{Y}$ belonging to semantic category $c(y) \in \mathcal{C}$, the scalar output $r(y) \in \mathbb{R}$ emitted by the proxy reward model admits an additive decomposition:

$$r(y) = q(y) + b_{c(y)}, \tag{5}$$

where $q(y) \in \mathbb{R}$ denotes the latent, instance-level intrinsic quality of response y dictated by its structural correctness and reasoning depth, and $b_{c(y)} \in \mathbb{R}$ represents a fixed-effect systemic bias specific to category $c(y)$, invariant to the microscopic token sequences of y.

To benchmark category-level behavior across the semantic space, we introduce a balanced reference dataset $\mathcal{D}_{\text{ref}}$. For each category $k \in \mathcal{C}$, the category-conditional expected reward is defined as $\mu_k \triangleq \mathbb{E}_{y \sim \mathcal{D}_{\text{ref}}}[r(y) \mid c(y) = k]$. By linearity of the conditional expectation applied to Definition 1, we have the exact identity:

$$\mu_k = \bar{q}_k + b_k, \tag{6}$$

where $\bar{q}_k \triangleq \mathbb{E}_{y \sim \mathcal{D}_{\text{ref}}}[q(y) \mid c(y) = k]$ characterizes the baseline quality of category k on the reference set. Across the discrete category universe $\mathcal{C}$, the macro-average expectation $\bar{\mu}$ and the inter-category standard deviation σ_μ are defined by:

$$\bar{\mu} \triangleq \frac{1}{C} \sum_{k=1}^{C} \mu_k, \qquad \sigma_\mu \triangleq \sqrt{\frac{1}{C} \sum_{k=1}^{C} (\mu_k - \bar{\mu})^2}. \tag{7}$$

This induces the dimensionless, standardized category deviation score for category k:

$$\varphi(k) \triangleq \frac{\mu_k - \bar{\mu}}{\sigma_\mu}. \tag{8}$$

By algebraic construction, $\varphi(k)$ possesses zero sample mean and unit sample variance over $\mathcal{C}$, serving as a scale-invariant metric of category preference.

In standard GRPO training, given a prompt $x \sim \mathcal{X}$, the policy π_θ generates an ensemble of K responses denoted by the ordered tuple $G = (y_1, y_2, \ldots, y_K)$. For each sample index $i \in \{1, \ldots, K\}$, we abbreviate the instance scalar reward as $r_i \triangleq r(y_i)$, its intrinsic quality as $q_i \triangleq q(y_i)$, and its semantic label as $c_i \triangleq c(y_i) \in \mathcal{C}$. The group sample mean $\bar{r}_G$ and sample standard deviation $s(G)$ are evaluated as:

$$\bar{r}_G \triangleq \frac{1}{K} \sum_{j=1}^{K} r_j, \qquad s(G) \triangleq \sqrt{\frac{1}{K} \sum_{j=1}^{K} (r_j - \bar{r}_G)^2}. \tag{9}$$

The standardized advantage assigned to response y_i by standard GRPO is defined as:

$$A_i \triangleq \frac{r_i - \bar{r}_G}{s(G)}. \tag{10}$$

A.2 Exact Dual-Channel Advantage Decomposition

To characterize the precise influence of category bias in non-asymptotic settings where K is typically small (such as $K = 4$ or $K = 8$), we bypass any heuristic statistical approximations and establish an exact algebraic decomposition.

For any candidate response $y_i \in G$, we define its individual quality deviation from its category expectation as $\delta q_i \triangleq q_i - \bar{q}_{c_i}$. Across the sampled group G, we define the empirical average of individual quality deviations as $\overline{\delta q}_G \triangleq \frac{1}{K}\sum_{j=1}^{K} \delta q_j$, and the empirical average of sampled category expectations as $\bar{\mu}_G \triangleq \frac{1}{K}\sum_{j=1}^{K} \mu_{c_j}$.

Proposition 1 (Dual-Channel Advantage Identity)**.** *For an arbitrary prompt x and any sampled group $G = (y_1, \ldots, y_K)$ of size $K \geq 2$, the standard GRPO advantage A_i assigned to sample y_i decomposes identically as:*

$$A_i = A_i^{\text{intra}} + A_i^{\text{inter}}, \tag{11}$$

where the intra-category quality competition channel and inter-category category-mean reward channel are defined respectively as:

$$A_i^{\text{intra}} \triangleq \frac{\delta q_i - \overline{\delta q}_G}{s(G)}, \qquad A_i^{\text{inter}} \triangleq \frac{\mu_{c_i} - \bar{\mu}_G}{s(G)}. \tag{12}$$

Proof. From the identity $\mu_{c_i} = \bar{q}_{c_i} + b_{c_i}$, we express the category bias directly as $b_{c_i} = \mu_{c_i} - \bar{q}_{c_i}$. Substituting this into Definition 1 for sample y_i yields:

$$r_i = q_i + b_{c_i} = q_i + (\mu_{c_i} - \bar{q}_{c_i}) = (q_i - \bar{q}_{c_i}) + \mu_{c_i}. \tag{13}$$

Recalling the definition $\delta q_i = q_i - \bar{q}_{c_i}$, each observed scalar reward is equivalently expressed as:

$$r_i = \delta q_i + \mu_{c_i}. \tag{14}$$

Averaging Equation 14 across all K instances within group G and leveraging the finite linearity of summations:

$$\bar{r}_G = \frac{1}{K}\sum_{j=1}^{K} r_j = \frac{1}{K}\sum_{j=1}^{K}(\delta q_j + \mu_{c_j}) = \left(\frac{1}{K}\sum_{j=1}^{K}\delta q_j\right) + \left(\frac{1}{K}\sum_{j=1}^{K}\mu_{c_j}\right) = \overline{\delta q}_G + \bar{\mu}_G. \tag{15}$$

Subtracting Equation 15 from Equation 14 yields the centered reward deviation:

$$r_i - \bar{r}_G = (\delta q_i + \mu_{c_i}) - (\overline{\delta q}_G + \bar{\mu}_G) = (\delta q_i - \overline{\delta q}_G) + (\mu_{c_i} - \bar{\mu}_G). \tag{16}$$

Dividing both sides by the group sample dispersion $s(G)$ gives:

$$\frac{r_i - \bar{r}_G}{s(G)} = \frac{\delta q_i - \overline{\delta q}_G}{s(G)} + \frac{\mu_{c_i} - \bar{\mu}_G}{s(G)}. \tag{17}$$

Recognizing the left-hand side as A_i and the right-hand terms as A_i^{intra} and A_i^{inter} completes the proof. □

Proposition 1 provides foundational insight into the pathological vulnerability of GRPO. The advantage function consists of two distinct components. The intra-category channel A_i^{intra} captures genuine capability, quantifying whether instance y_i outperforms its category peers in terms of micro-level generation. Conversely, the inter-category channel A_i^{inter} serves as a systemic baseline offset. When a category suffers from inflated bias b_{c_i}, its reference mean μ_{c_i} dominates the group baseline $\bar{\mu}_G$. As a result, sub-optimal responses from positively biased categories receive spurious positive advantage scores solely by virtue of their category affiliation, creating an optimization loop that triggers mode collapse.

A.3 Structural Scale Compatibility of the Penalty Term

To suppress the unearned systemic channel A_i^{inter}, one must formulate a calibrated reward $r'_i = r_i - \Delta_i$. We consider a generalized penalty parameterized by $\Delta_i \triangleq f(G) \cdot \phi(c_i)$, where $f(G)$ denotes a group-dependent scale factor and $\phi(c_i)$ represents a category metric.

For the purpose of theoretical analysis, we define the fixed-dispersion normalized score:

$$\widetilde{A}_i \triangleq \frac{r'_i - \bar{r}'_G}{s(G)}, \tag{18}$$

which isolates the first-order effect of reward calibration on the group-relative numerator while holding the empirical dispersion $s(G)$ fixed. In actual deployment, the calibrated GRPO advantage evaluates as $A'_i \triangleq (r'_i - \bar{r}'_G)/s(G')$, where $s(G') \triangleq \sqrt{\frac{1}{K}\sum_j (r'_j - \bar{r}'_G)^2}$. When the variance shift induced by calibration is well-behaved, $\widetilde{A}_i$ serves as a direct first-order surrogate for A'_i.

Proposition 2 (Scale Compatibility and Non-Singularity under Collapse)**.** *Let $\delta A_i \triangleq \widetilde{A}_i - A_i$ denote the net advantage displacement under the fixed-dispersion score. In order for the calibration mechanism to satisfy:*

1. *Scale compatibility, such that δA_i is strictly dimensionless and invariant to linear rescalings of the underlying reward metric;*
2. *Non-singularity under mode collapse, such that in the event of severe intra-group homogenization where sample variance vanishes ($s(G) \to 0$), the advantage perturbation remains uniformly bounded ($\lim_{s(G)\to 0} |\delta A_i| < \infty$);*

setting $f(G) = \beta \cdot s(G)$ for some fixed scalar $\beta > 0$ and choosing $\phi(c)$ as the standardized deviation $\varphi(c)$ represents a natural and robust formulation.

Proof. Defining the empirical mean of the penalty terms over group G as $\bar{\Delta}_G \triangleq \frac{1}{K}\sum_{j=1}^K \Delta_j$, the calibrated group mean reward is:

$$\bar{r}'_G = \frac{1}{K}\sum_{j=1}^{K}(r_j - \Delta_j) = \bar{r}_G - \bar{\Delta}_G. \tag{19}$$

The fixed-dispersion normalized score takes the form:

$$\widetilde{A}_i = \frac{(r_i - \Delta_i) - (\bar{r}_G - \bar{\Delta}_G)}{s(G)} = \frac{r_i - \bar{r}_G}{s(G)} - \frac{\Delta_i - \bar{\Delta}_G}{s(G)}. \tag{20}$$

Subtracting $A_i = (r_i - \bar{r}_G)/s(G)$, the exact advantage displacement simplifies to:

$$\delta A_i = -\frac{\Delta_i - \bar{\Delta}_G}{s(G)}. \tag{21}$$

Substituting $\Delta_i = f(G) \cdot \phi(c_i)$ and defining $\bar{\phi}_G \triangleq \frac{1}{K}\sum_{j=1}^K \phi(c_j)$, we have:

$$\delta A_i = -\frac{f(G)\cdot\phi(c_i) - \frac{1}{K}\sum_{j=1}^K f(G)\cdot\phi(c_j)}{s(G)} = -\left(\frac{f(G)}{s(G)}\right)\big(\phi(c_i) - \bar{\phi}_G\big). \tag{22}$$

If $f(G)$ is chosen independent of $s(G)$ (e.g., setting $f(G) = \beta$), the ratio exhibits an asymptotic explosion as homogenization occurs:

$$\lim_{s(G)\to 0} |\delta A_i| = \lim_{s(G)\to 0} \frac{\beta|\phi(c_i) - \bar{\phi}_G|}{s(G)} = \infty. \tag{23}$$

This induces severe numerical instability and gradient explosion during training. Conversely, choosing $f(G) = \beta \cdot s(G)$ leads to exact algebraic cancellation of the dispersion parameter:

$$\delta A_i = -\frac{\beta \cdot s(G)}{s(G)}\big(\phi(c_i) - \bar{\phi}_G\big) = -\beta\big(\phi(c_i) - \bar{\phi}_G\big). \tag{24}$$

Consequently, $\lim_{s(G)\to 0} |\delta A_i| = \beta|\phi(c_i) - \bar{\phi}_G| < \infty$, ensuring uniform stability across all optimization regimes. Setting $\phi(c) \triangleq \varphi(c)$ guarantees scale compatibility. □

A.4 Linear Contraction of the Category-Mean Channel

We now examine how the established calibration term $\Delta_i = \beta \cdot s(G) \cdot \varphi(c_i)$ modulates the individual components of the advantage decomposition established in Proposition 1.

Lemma 1 (Exact Group-Centering of Standardized Scores)**.** *Let $\bar{\varphi}_G \triangleq \frac{1}{K}\sum_{j=1}^{K}\varphi(c_j)$ denote the group average of normalized category deviations. The centered deviation strictly satisfies:*

$$\varphi(c_i) - \bar{\varphi}_G = \frac{\mu_{c_i} - \bar{\mu}_G}{\sigma_\mu}. \tag{25}$$

Proof. Substituting Equation 8 into the group mean expression:

$$\bar{\varphi}_G = \frac{1}{K}\sum_{j=1}^{K}\left(\frac{\mu_{c_j} - \bar{\mu}}{\sigma_\mu}\right) = \frac{1}{\sigma_\mu}\left(\frac{1}{K}\sum_{j=1}^{K}\mu_{c_j} - \frac{1}{K}\sum_{j=1}^{K}\bar{\mu}\right) = \frac{\bar{\mu}_G - \bar{\mu}}{\sigma_\mu}. \tag{26}$$

Subtracting this from $\varphi(c_i) = (\mu_{c_i} - \bar{\mu})/\sigma_\mu$, the common constant term $\bar{\mu}$ cancels directly:

$$\varphi(c_i) - \bar{\varphi}_G = \frac{(\mu_{c_i} - \bar{\mu}) - (\bar{\mu}_G - \bar{\mu})}{\sigma_\mu} = \frac{\mu_{c_i} - \bar{\mu} - \bar{\mu}_G + \bar{\mu}}{\sigma_\mu} = \frac{\mu_{c_i} - \bar{\mu}_G}{\sigma_\mu}, \tag{27}$$

which completes the proof. □

Theorem 2 (Linear Contraction of the Category-Mean Channel)**.** *Under the calibrated reward formulation $r'_i = r_i - \beta \cdot s(G) \cdot \varphi(c_i)$, the fixed-dispersion normalized score $\widetilde{A}_i$ satisfies the exact algebraic identity:*

$$\widetilde{A}_i = \underbrace{\frac{\delta q_i - \overline{\delta q}_G}{s(G)}}_{A_i^{\text{intra}}} + \left(1 - \frac{\beta \cdot s(G)}{\sigma_\mu}\right)\underbrace{\frac{\mu_{c_i} - \bar{\mu}_G}{s(G)}}_{A_i^{\text{inter}}}. \tag{28}$$

Proof. Combining the exact decomposition from Proposition 1 with the net displacement from Proposition 2, we write:

$$\widetilde{A}_i = A_i + \delta A_i = \left(\frac{\delta q_i - \overline{\delta q}_G}{s(G)} + \frac{\mu_{c_i} - \bar{\mu}_G}{s(G)}\right) - \beta\big(\varphi(c_i) - \bar{\varphi}_G\big). \tag{29}$$

Applying Lemma 1 to substitute $\varphi(c_i) - \bar{\varphi}_G$:

$$\widetilde{A}_i = \frac{\delta q_i - \overline{\delta q}_G}{s(G)} + \frac{\mu_{c_i} - \bar{\mu}_G}{s(G)} - \frac{\beta}{\sigma_\mu}(\mu_{c_i} - \bar{\mu}_G). \tag{30}$$

Factoring out the term $(\mu_{c_i} - \bar{\mu}_G)$ from the second and third components yields:

$$\widetilde{A}_i = \frac{\delta q_i - \overline{\delta q}_G}{s(G)} + \left(\frac{1}{s(G)} - \frac{\beta}{\sigma_\mu}\right)(\mu_{c_i} - \bar{\mu}_G). \tag{31}$$

Extracting the factor $1/s(G)$ from the bracketed expression:

$$\left(\frac{1}{s(G)} - \frac{\beta}{\sigma_\mu}\right)(\mu_{c_i} - \bar{\mu}_G) = \left(1 - \frac{\beta \cdot s(G)}{\sigma_\mu}\right)\left(\frac{\mu_{c_i} - \bar{\mu}_G}{s(G)}\right). \tag{32}$$

Substituting this back yields Equation 28, completing the proof. □

Theorem 2 reveals the dual benefits of our calibration operator: First, the intra-category quality competition channel A_i^{intra} retains a multiplier strictly equal to unity, ensuring that micro-level relative generation quality is preserved without attenuation. Second, the category-mean channel A_i^{inter} is scaled by the contraction factor $\gamma(G) \triangleq 1 - \beta \cdot s(G)/\sigma_\mu$, where $\gamma(G) \in (0, 1)$ under standard hyperparameter configurations. Compared to enforcing hard cancellation ($\gamma(G) = 0$), which requires $\beta \propto 1/s(G)$ and risks asymptotic instability as $s(G) \to 0$, soft shrinkage preserves training stability while maintaining meaningful exploration over legitimate category-level quality variations.

B EXPERIMENTAL SETUP

Overview. We evaluate REI on three sources of performance-oriented evolution traces: an internal industrial corpus from a large-scale production recommender system, a public benchmark constructed from modded-nanogpt speedrun records, and a second public benchmark based on rustc-perf (Appendix G). In all three settings, each sample links a pre-change system state, a performance-motivated modification, and an observed post-change outcome. The full data construction and evaluation pipelines are programmatic, with no human intervention in sample selection, split assignment, or metric computation.

Industrial corpus. The industrial corpus contains 60,000+ recommender-system evolution samples, covering success/failure annotations of changes from 24 code repositories. Among them, 54,710 samples are used for training and 6,291 for testing.

Each raw industrial record corresponds to one recommender-system evolution attempt and contains three parts: (1) an engineering-code snapshot, including the full training repository at submission time; (2) training logs and metrics, including loss curves and downstream evaluation outputs; and (3) experiment metadata, such as repository identifiers, commit hash, experiment id, timestamps, runner, model name, dataset identifier, and checkpoint path. This structure matches the real evolution loop in production: a candidate direction is first implemented, then trained and evaluated, and only afterwards can its value be determined.

Industrial data construction. To reduce sparsity, we form pairwise samples from historical experiment trajectories rather than keeping only strictly adjacent changes. To reduce noise, we apply a fully automatic two-stage filtering pipeline.

The first stage is execution-logic filtering. We perform reachability analysis from the script entry point and discard changed functions or classes that are not reachable from actual execution paths. We then parse Python files into abstract syntax trees (ASTs), remove debugging statements and comments, normalize import ordering, and suppress purely superficial edits. After normalization, we compute pairwise code differences and apply configuration whitelisting so that engineering variables such as output directories, resource settings, tracing options, and data endpoints are removed, while performance-relevant fields such as optimizer, learning rate, weight decay, batch size, model structure, and feature configuration are retained.

The second stage is semantic intent filtering. The purified code delta is mapped into explicit evolution intent by a fixed extraction pipeline. Modifications whose intent cannot be reliably recovered are discarded as ambiguous noise. The retained samples are further organized into structured direction annotations used by the generator and by the diversity analysis.

NanoGPT benchmark. The nanoGPT benchmark is built from the modded-nanogpt speedrun leaderboard (`KellerJordan/modded-nanogpt`), using pull requests and associated repository history from 2024-05 to 2026-01. We use two leaderboard tracks: track 1 (GPT-2 Small) and track 2 (GPT-2 Medium). We choose this source because it naturally records repeated performance-oriented iteration: contributors propose a change, realize it in code, run the full training/evaluation pipeline, and report the resulting gain. Compared with generic open-source repository histories, these records are cleaner for ideation learning because the dominant intent is leaderboard improvement rather than maintenance, refactoring, or feature development.

Each retained public item is annotated with one of five direction families: C1 system/kernel, C2 optimizer, C3 architecture, C4 attention, and C5 loss. On the public benchmark, these categories are the ground-truth direction families provided by the dataset and are used for method-level evaluation.

NanoGPT data curation and split. The raw nanoGPT library contains 190 candidate records. After removing invalid entries, we retain 107 valid ideas for the benchmark. Retained items are labeled by observed outcome: *strong* for realized speedup $\geq 1\%$ and *null* for valid attempts without meaningful gain. Invalid records are excluded from the curated set. We train on 79 track-1 items and evaluate on a 28-question held-out test set, of which 23 have ground-truth labels for family-level evaluation. The split is defined at the PR level to prevent leakage, so all artifacts associated with a held-out PR are kept out of training.

Prompt construction. Each prompt contains the current scenario context together with up to 8 historical changes retrieved from the corresponding training library. The model is asked to generate a compact single-line proposal in the format `[CATEGORY] <what to change>`.

In the public benchmark, the ground-truth label is the accepted direction family attached to the held-out record. In the industrial setting, where no ground-truth taxonomy exists, family labels used for diversity analysis are obtained by the same LLM extraction pipeline used during industrial data construction.

Class balance. To reduce trivial under-exposure of rare families, SFT up-samples minority categories in the public training set. We additionally tested whether the observed semantic collapse could be explained by family-frequency skew in the data or reward pipeline. It could not: the collapse remains qualitatively unchanged after category rebalancing, indicating that the phenomenon is not an artifact of popularity bias alone.

Implementation. The nanoGPT benchmark is relatively small, so nanoGPT-benchmark experiments use `Qwen2.5-0.5B` as the default backbone and run on 2 NVIDIA A100 80GB GPUs with standard data parallelism. By contrast, the industrial corpus is much larger and the industrial experiments use `Qwen3-8B` on 8 NVIDIA A100 80GB GPUs. Unless otherwise stated, SFT, reward-model training, and GRPO follow the same stage definitions described in the main paper, with hardware scaled to match the dataset size and backbone.

Rollout sampling uses temperature 0.99, top-$p = 0.99$, and top-$k = 100$. Megatron parallelism is disabled beyond data parallelism (TP=PP=EP=1). Software versions are PyTorch 2.6.0, vLLM 0.8.4, and Transformer-Engine 2.2.0.

The main-text nanoGPT-benchmark tables report a single representative seed (42); five-seed means over $\{42, 211, 985, 43, 44\}$ and paired significance tests are given in Appendix L. All post-training baselines share the same SFT initialization, reward model, training prompts, and test set, and differ only in the RL objective or diversity-preserving mechanism.

Evaluation protocol. For each test prompt, the model generates 8 candidates unless otherwise specified, ranked by the corresponding reward model. Candidate-set quality uses four metrics. Hit@1 is the fraction of prompts whose top-ranked candidate falls in a correct method family. Hit@8 is the fraction for which at least one of the 8 candidates falls in a correct family. Coverage is, for prompts with multiple valid families, the fraction of those families covered by the candidate set. TV to Reference Prior is the total variation distance between the generated method-category distribution and the reference-corpus prior, where smaller is better. In the industrial setting, each evaluation instance has a single ground-truth family, so Coverage is identical to Hit@8. In the public benchmark, a prompt may have multiple valid families because several distinct PR directions from the same repository state are treated as valid targets, so Coverage and Hit@8 need not coincide.

Method-level diversity uses four metrics. Distinct Method Count is the number of distinct families among the 8 candidates. Top-Family Share is the share of the most frequent family. Max Within-Family Count is the largest number of candidates in a single family. Single-Family Collapse Rate is the fraction of prompts whose 8 candidates all fall in one family. Cross-task evaluates transfer beyond the training task, while long-tail removes the dominant head family before recomputing the metrics, testing whether a method can still recover valid non-head families.

Fairness of comparison. All compared post-training baselines are trained from the same SFT checkpoint and evaluated on the same held-out prompts with the same candidate budget. For industrial offline evaluation, the same system context and candidate budget are given to all model baselines. All model outputs are processed under the same ranking and metric pipeline, with no manual intervention or example-specific adjustment. Human-expert comparisons use the same task description and access to the same scenario context. The method-category signal is provided to all compared methods as a shared input format. Baselines receive the same [CATEGORY]-tagged prompts but retain their original objectives. SPOD differs in that it uses this shared signal for category-aware reward correction and top-1 retention.

C Protocols for the Diversity-Collapse Analysis

Evaluation data and slices. Figures 2 and 3 use the public nanoGPT-speedrun benchmark with the 79-item training split and 28-prompt held-out test split, with Qwen2.5-0.5B as the ideation policy. Ten late-stage prompts from track 1 form the same-task slice and evaluate temporal extrapolation within the training track. The remaining 18 prompts come from track 2 and form the cross-task slice, evaluating transfer to a track unseen during training. Both slices are fully held out from policy training.

Reward and long-tail summaries. For each prompt, the policy generates eight candidates, ranked by the same verifier used during training. In Figure 2(a), mean reward is computed by averaging verifier scores over the eight candidates and then over prompts. Best-of-8 reward first takes the highest verifier score among the eight candidates for each prompt and then averages across prompts. The former summarizes overall candidate-set quality, whereas the latter measures whether sampling still produces at least one high-scoring direction.

For Figure 2(b), generated candidates are mapped to the five speedrun method families: system/kernel, optimizer, architecture, attention, and loss. Families are grouped by generated share into head ($\geq 40\%$), mid (10%–40%), and tail ($< 10\%$), after which Hit@1 is recomputed within each bucket. Prompts without a usable ground-truth family label are excluded from Hit@1. This analysis tests whether post-training gains are broadly distributed or concentrated in a small number of dominant families.

Surface- and method-level diversity. Figure 3 separates surface-level from method-level changes. The left panel reports token-level entropy during generation. The middle panel tracks three method-level statistics normalized to step 0: category entropy, the share of the most frequent family, and the number of distinct families n_{cat} covered by the eight candidates. The right panel reports final-checkpoint percentage changes relative to the SFT initialization. Surface statistics are computed directly from generated text: word-level Distinct-2 measures bigram repetition, the TF–IDF Vendi score measures the effective number of distinct texts, and mean pairwise cosine distance measures separation in TF–IDF space. Method-level statistics use the offline family annotations. All percentage changes use the same 28×8 evaluation batch for the method being compared with its SFT initialization.

Verifier-fidelity intervention. Figure 4 studies whether stronger verifier agreement is associated with greater method-level concentration. Verifier fidelity is defined as pairwise ranking accuracy between verifier scores and realized gains, where 0.5 denotes random ranking and 1.0 perfect agreement. Realized gains come from the nanoGPT-speedrun submission history. For each submission, we compute the relative wall-time change from the adjacent record and apply same-era quantile normalization to reduce temporal drift. This reduces the Spearman correlation between record index and normalized gain from -0.46 to -0.08. Historical submissions are grouped into strong, null, and reject outcomes using the community significance threshold $p < 0.01$.

To vary fidelity while holding the remaining setup fixed, we keep the deployed verifier, policy initialization, training data, model scale, and optimization configuration unchanged and perturb only the verifier score:

$$s_\lambda = s_{\mathrm{RM}} + \lambda\epsilon, \qquad \epsilon \sim \mathcal{N}(0,1), \tag{33}$$

where λ controls the noise magnitude. The weak, medium, and strong conditions have measured pairwise ranking accuracies of 0.55, 0.67, and 0.83, respectively. After training under each condition, concentration is measured as the final share of generated candidates assigned to the most frequent method family. The corresponding shares are 49.1%, 58.8%, and 62.5%.

D Baseline Methods

We compare SPOD with supervised learning, reward-driven policy optimization, and token-level exploration controls. For a prompt x, let y_i be a sampled response, $s_{i,t} = (x, y_{i,<t})$ its prefix, R_i its reward, and $\rho_{i,t} = \pi_\theta(y_{i,t} \mid s_{i,t})/\pi_{\mathrm{old}}(y_{i,t} \mid s_{i,t})$ the importance ratio. Write $H_{i,t}$ for the entropy of the full next-token distribution at that prefix.

Supervised fine-tuning (SFT). SFT maximizes the likelihood of reference responses, minimizing $-\mathbb{E}_{(x,y)\sim\mathcal{D}} \log \pi_\theta(y \mid x)$. In REI, repository-derived examples provide supervision for learning reusable ideation patterns. SFT provides a reference before reward optimization, without explicitly optimizing candidate-set coverage.

Group Relative Policy Optimization (GRPO). GRPO generates multiple responses per prompt and computes $A_i = (R_i - \bar{R})/(\mathrm{std}(R) + \epsilon)$ within each group, avoiding a separate value model. Its clipped policy surrogate averages $\min\{\rho_{i,t}A_i, \mathrm{clip}(\rho_{i,t}, 1-\epsilon_l, 1+\epsilon_h)A_i\}$ over tokens and responses. It reinforces candidates with above-average rewards.

Entropy regularization (Ent. Reg.). This baseline augments reward optimization with an entropy bonus, $J_{\mathrm{RL}} + \alpha\mathbb{E}_{i,t}[H_{i,t}]$, where $\alpha > 0$ controls the incentive to retain uncertainty. It discourages premature concentration of next-token probabilities. However, preserving token entropy does not directly ensure that generated ideas span distinct methods: different wordings can still express the same solution family.

Entropy-based advantage adjustment (Entro. Adv.). These variants incorporate conditional entropy into the advantage or its weighting, making policy updates depend on both reward feedback and uncertainty at a generation step. Unlike an additive entropy regularizer, they modify the learning signal attached to sampled tokens. Such interventions aim to preserve exploration at uncertain positions, but uncertainty alone does not determine whether a particular update increases or decreases entropy.

Clipped IS-weight Policy Optimization (CISPO). CISPO clips importance weights while retaining a log-likelihood gradient:

$$J_{\mathrm{CISPO}} = \mathbb{E}\left[\frac{1}{\sum_i |y_i|}\sum_{i,t} \mathrm{sg}\big[\mathrm{clip}(\rho_{i,t}, 1-\epsilon_l^{\mathrm{IS}}, 1+\epsilon_h^{\mathrm{IS}})\big]\, A_i \log \pi_\theta(y_{i,t} \mid s_{i,t})\right],$$

where sg stops gradients through the weight. Unlike the clipped GRPO surrogate, exceeding a ratio threshold does not itself eliminate the token's direct policy gradient. This preserves learning signals while limiting their importance weights; it does not recalibrate the underlying reward or identify redundant method families.

STEER. STEER estimates the local entropy change induced by a policy update from token probabilities, advantages, and conditional entropy. It attenuates updates with large predicted changes using $\lambda_{i,t} = \exp(-k|\Omega_{i,t}|)$, where $\Omega_{i,t}$ estimates entropy change and k sets the minimum weight. Thus it regulates the rate of entropy change, rather than simply maximizing entropy, and suppresses both abrupt concentration and excessive dispersion.

Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO). DAPO combines asymmetric ratio clipping, dynamic sampling of informative response groups, token-level loss aggregation, and overlong-response reward shaping. A higher upper clipping threshold gives positively rewarded, initially unlikely tokens more room to grow; dynamic sampling excludes groups with identical rewards and no relative learning signal. These baselines test whether generic exploration controls suffice to preserve method diversity, compared with SPOD's category-aware reward correction.

E Parameter Sensitivity

Debiasing strength. The debiasing strength β controls the reward correction, with $\beta = 0$ reducing exactly to GRPO. We sweep $\beta \in \{0, 0.5, 1.0, 1.5\}$ and find that SPOD is robust across a broad range of values: all nonzero choices of β outperform the $\beta = 0$ GRPO baseline overall across the eight metrics in Figure 6. Among them, $\beta = 1.0$ gives the best overall trade-off and is therefore used in all other experiments. Increasing β further to 1.5 causes some metrics to regress, suggesting that overly strong correction can start to suppress genuinely strong in-family candidates, although it still remains better overall than GRPO.

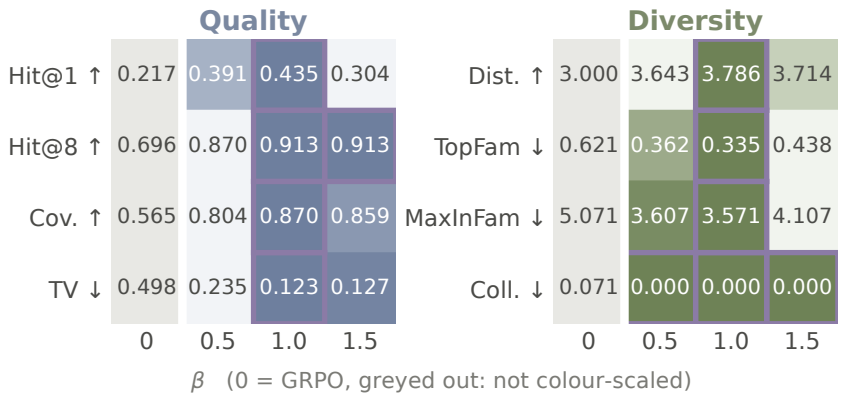

Figure 6: Sensitivity to the debiasing strength β. Abbreviations follow Table 5. The $\beta = 0$ (GRPO) column is greyed out and excluded from the per-row min–max scaling; raw values are shown and the best value in each row is outlined.

Exemption depth. We further ablate the exemption depth k, i.e. how many top-ranked in-family candidates are shielded from the penalty. Tightening k from 3 to 1 improves all eight metrics monotonically (Figure 7): a larger k shields more of the in-family head and thus dilutes the debiasing signal, while $k = 1$ is the tightest setting that still protects the strongest in-family candidate. We use $k = 1$ in the remaining experiments.

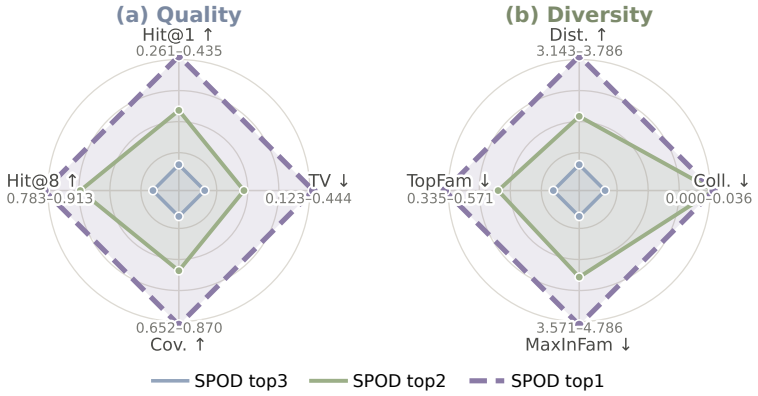

Figure 7: Sensitivity to top-k exemption. Axes are min–max scaled across the tested settings; outward is better. Metric abbreviations follow Table 5.

Default setting across datasets. Unless otherwise stated, all reported experiments on the industrial system, nanoGPT, and rustc-perf use the same SPOD hyperparameters, namely $\beta = 1.0$ and $k = 1$. We do not tune these parameters separately for each dataset.

F Industrial Cross-Task and Long-Tail Evaluation

Evaluation scope. We report additional industrial offline evaluations on two slices: cross-task, which assesses generalization beyond the training task, and long-tail, which evaluates performance on long-tail method categories. Table 6 reports the four quality metrics for all eight methods. Each industrial instance has a single ground-truth method category, so Coverage equals Hit@8 for every method: covering the ground truth is equivalent to having at least one matching candidate. Bold entries indicate column-wise best values.

Table 6: Industrial offline quality on cross-task and long-tail slices. Coverage equals Hit@8 because each instance has one ground-truth category. Higher is better except for TV to Reference Prior.

Method	Cross-Task				Long-Tail			
	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Hit@1↑	Hit@8↑	Cov.↑	TV↓
SFT	0.500	0.885	0.885	0.112	0.248	0.426	0.426	0.093
GRPO	0.495	0.825	0.825	**0.069**	0.278	0.422	0.422	0.087
STEER	0.500	0.890	0.890	0.104	0.300	0.440	0.440	0.075
DAPO	0.515	0.900	0.900	0.229	0.314	0.441	0.441	0.168
Ent. Reg.	0.490	0.890	0.890	0.117	0.266	0.429	0.429	0.209
CISPO	0.475	0.849	0.849	0.203	0.255	0.502	0.502	0.191
Entro. Adv.	0.519	0.882	0.882	0.303	0.313	0.497	0.497	**0.068**
SPOD	**0.579**	**0.916**	**0.916**	0.085	**0.387**	**0.570**	**0.570**	0.080

Cross-task results. SPOD achieves the highest Hit@1 (0.579), Hit@8 (0.916), and Coverage (0.916). Relative to the best competing baseline for each metric, its Hit@1 exceeds Entro. Adv. by 0.060 (6.0 percentage points), while its Hit@8 and Coverage exceed DAPO by 0.016 (1.6 percentage points).

Long-tail results. SPOD again attains the highest Hit@1 (0.387), Hit@8 (0.570), and Coverage (0.570). Its Hit@1 exceeds DAPO, the strongest alternative on this metric, by 0.073 (7.3 percentage points); its Hit@8 and Coverage exceed CISPO by 0.068 (6.8 percentage points). For TV, Entro. Adv. ranks first (0.068), followed by STEER (0.075) and SPOD (0.080). These results support improved recovery of valid long-tail methods, while indicating that candidate-level accuracy and agreement with the reference category distribution capture different aspects of performance.

G Generalization to a Second Public Benchmark: rustc-perf

The main paper evaluates SPOD in two application scenarios: an industrial recommender system (*AI for Recommendation*) and the nanoGPT speedrun (*AI for AI*). To test whether the quality–diversity behavior of SPOD transfers to a qualitatively different scenario, we evaluate SPOD in a third setting—*AI for Software Engineering*—using a second, fully public benchmark drawn from the Rust compiler ecosystem. In this task, given the recent optimization history of the compiler, the model proposes one next change that may reduce compilation time on the rustc-perf benchmark suite. The corpus contains $1{,}591$ landed changes collected from public rustc-perf triage reports and rust-lang/rust pull requests, and a strict temporal split yields $1{,}414$ training states and 177 held-out test states. At each state the model observes summary statistics from the preceding 52 weeks and the eight most recent compiler changes, then proposes one concrete optimization in the form "`[category] change description`". The six categories cover backend and code generation, type and trait solving, incremental queries, frontend parsing, the standard library, and infrastructure and tooling. Crucially, the gold utility is derived from the compile-time change measured by the public rustc-perf continuous-integration reports rather than from any model-generated judgment, so the evaluation does not rely on a private task or a learned gold signal.

Table 7: Summary of the public rustc-perf benchmark.

Item	Setting
Source	Public rustc-perf triage reports and rust-lang/rust pull requests
Base model	Qwen3-1.7B-Base
Task granularity	One state per landed compiler change
Dataset size	$1{,}591$ states: $1{,}414$ train / 177 temporal hold-out
Evaluation	$k = 8$ samples per state; 157 of 177 test states carry usable gold labels
Gold signal	Measured compile-time change from rustc-perf CI

We report the same eight metrics as in the main paper, which separate answer quality from category-level diversity. Hit@1, Hit@8, Coverage, and the Distinct Method Count (Dist.) are better when larger, while TV to the reference category distribution, Top-Family Share (TopFam), Max Within-Family Count (MaxInFam), and the Single-Family Collapse Rate (Coll.) are better when smaller. All methods are evaluated on the same temporal hold-out with the same reward model. Tables 8 and 9 report quality and diversity, respectively; bold marks the best result in each column.

Table 8: Quality results on the rustc-perf benchmark.

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓
SFT	0.236	0.803	0.727	0.226
GRPO	0.198	0.809	0.727	0.252
DAPO	0.229	0.764	0.702	0.270
Entro. Adv.	0.223	0.783	0.706	0.232
Ent. Reg.	0.210	0.803	0.732	0.235
CISPO	0.248	0.847	0.770	0.247
STEER	0.248	0.777	0.715	0.240
SPOD	**0.261**	**0.860**	**0.773**	**0.218**

Table 9: Category-diversity results on the rustc-perf benchmark. Coll. is the fraction of states for which all eight samples fall into one category.

Method	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SFT	**4.616**	0.233	**2.989**	**0.000**
GRPO	4.418	0.262	3.057	**0.000**
DAPO	4.463	0.260	3.096	**0.000**
Entro. Adv.	4.362	0.258	3.085	**0.000**
Ent. Reg.	4.514	0.259	3.209	**0.000**
CISPO	4.463	0.254	3.040	**0.000**
STEER	4.412	0.253	3.085	**0.000**
SPOD	4.537	**0.201**	3.051	**0.000**

Main finding. On this public compiler-optimization task, standard reinforcement learning increases category concentration relative to the SFT initialization: GRPO raises Top-Family Share from 0.233 to 0.262 while reducing the Distinct Method Count. SPOD reverses this trend, attaining the lowest Top-Family Share (0.201) together with the highest Hit@1 (0.261), the highest Hit@8 (0.860), the highest Coverage (0.773), and the lowest TV (0.218). These results reproduce the central quality–diversity claim of the main paper on an independent public benchmark whose gold signal comes from measured compile-time changes rather than from model-generated judgments.

H Multi-Round Replay Simulation: Details

This appendix expands the replay simulation summarized in Section 4.3. The core issue is not diversity in the abstract, but a concrete optimization behavior: under reward-only training, policies increasingly shift probability mass toward the head method family that is easiest to score well under the current reward model or dataset prior. This can preserve superficial single-round performance, but over repeated iteration it systematically removes non-head yet genuinely high-yield directions. We observe the same overall pattern on both the public speedrun data and the industrial dataset; Section H.2 reports the industrial replication.

H.1 Public Benchmark: Family-Level Concentration and Long-Horizon Stagnation

On the public speedrun data, we sort the historically accepted improvements within each method family by realized gain. At each round, a policy samples one family according to the trained model's

measured output distribution and receives the next unused real gain from that family. This setup asks a direct question: if a policy keeps most of its probability mass on a small number of families, does repeated reuse of the same high-prior direction eventually hurt long-horizon performance?

The replay results support this concern. GRPO concentrates heavily on the head family C4, allowing it to obtain some early reward from attention-related directions. However, its cumulative reward grows more slowly in later rounds. Under T=30, GRPO spends on average 8.96 rounds on families that no longer yield additional replay reward. By contrast, SPOD preserves broader coverage across C1–C5, so its replay distribution continues to draw reward from multiple families in later rounds (Figure 8). This pattern is consistent with SPOD's advantage over GRPO under short-horizon, long-horizon, and cross-task settings, and with the widening gap as the horizon grows.

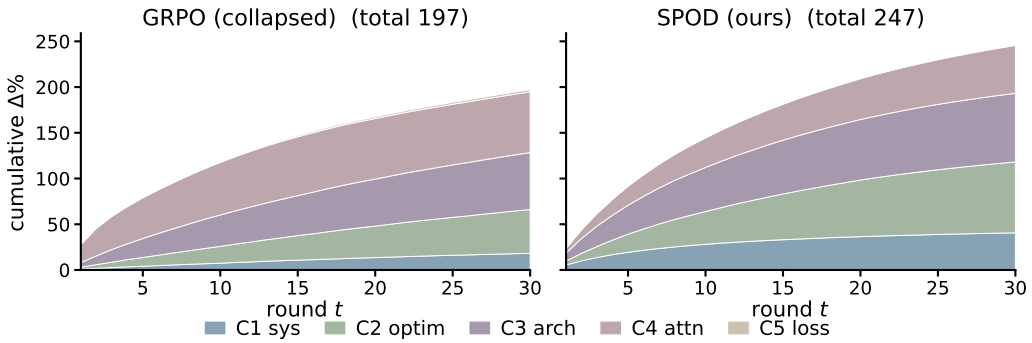

Figure 8: Family-decomposed cumulative reward on the public benchmark at T=30. GRPO (left) derives most of its reward from a single dominant family and grows more slowly later in the horizon. SPOD (right), by contrast, maintains broader coverage across C1–C5 and continues to accumulate reward from multiple families throughout the horizon.

The cross-task result supports the same interpretation. After the task pool switches at round 15, GRPO remains more concentrated on its previous head-family preference. Because SPOD preserves broader family coverage, its replay distribution continues to place probability mass on families that remain productive in the new task pool, allowing it to stay ahead after the switch (Figure 9).

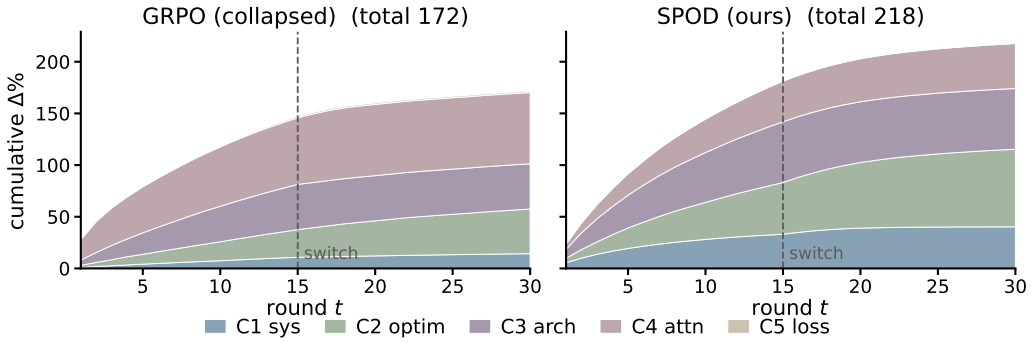

Figure 9: Cross-task replay on the nanoGPT benchmark. The task pool switches at round 15 (dashed line). GRPO remains concentrated on its previous head-family preference, whereas SPOD maintains broader coverage and retains probability mass on families that continue to yield reward after the switch.

H.2 Industrial Replication: Misalignment Between Popularity and Productivity

The industrial replay leads to the same conclusion, but the key pattern is a systematic misalignment between popularity and productivity. In the industrial library, the most common method family is not the family whose individual directions contribute the strongest returns. However, reward-only RL continues to push probability mass toward the most common and easiest-to-repeat family, rather than the families that contribute the largest long-horizon payoff.

This pattern is clear in the industrial replay (Figure 10). Loss design is the most frequent family in the historical library, and GRPO further increases its allocation to that family after RL. But the highest-payoff directions are concentrated more heavily in colder families such as optimization and training infrastructure. In other words, GRPO learns that repeatedly betting on the popular family is the easiest way to maintain high reward, not that this family contains the best directions. SPOD acts exactly on this failure mode: it weakens the one-sided concentration toward the head family and reallocates more mass toward optimization and training-infra, which leads to substantially higher cumulative reward over time.

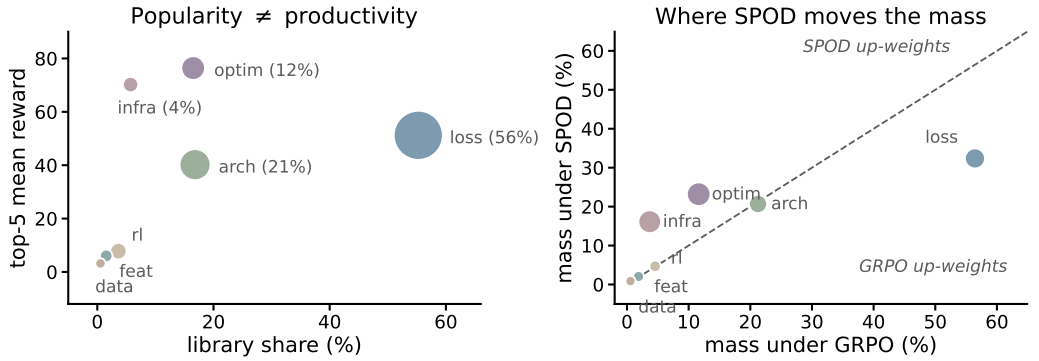

Figure 10: Popularity–productivity misalignment on the industrial library. Left: family popularity (share of historical library) versus productivity (mean realized gain); the most frequent family is not the most productive one. Right: the family allocation GRPO learns versus SPOD; marker area encodes GRPO's probability mass on that family and the largest-mass families are annotated, and markers above the dashed diagonal are up-weighted by SPOD relative to GRPO. SPOD shifts mass from the popular head family toward the higher-yield optimization and training-infra families.

The industrial results are fully consistent with this mechanism. SPOD outperforms GRPO under short-horizon, long-horizon, and cross-task settings, and comes much closer to the oracle upper bound in the long-horizon setting. This gain is not driven by a few isolated lucky samples. It comes from a persistent difference in family allocation: GRPO obtains too much of its cumulative reward from the head family, while SPOD recovers substantially more useful directions from optimization and training-infra, yielding both higher average reward per round and higher final cumulative reward (Figure 11).

The cross-task result supports the same explanation. After the task pool switches, both methods encounter new high-payoff directions, but SPOD retains its advantage because its higher allocation to optimization and training-infra continues to transfer to the new task (Figure 12). Its gain therefore does not depend on any one fixed family, but on a healthier long-horizon allocation policy.

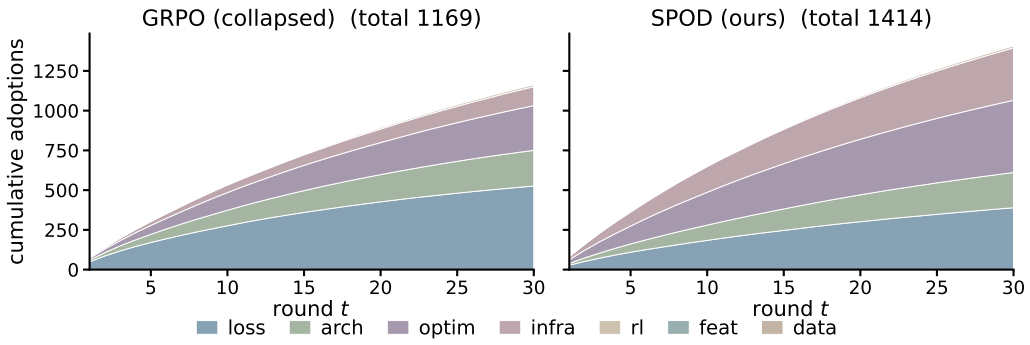

Figure 11: Cumulative reward over the horizon on the industrial replay. SPOD stays above GRPO throughout the horizon. This advantage is consistent with a more balanced allocation across method families in the replay.

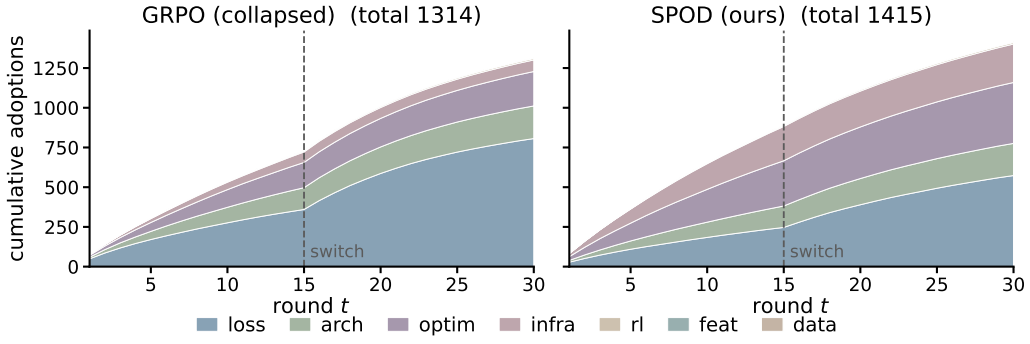

Figure 12: Cross-task replay on the industrial dataset. After the task pool switches, SPOD remains ahead and continues to accumulate reward from a broader mix of method families. By contrast, GRPO remains heavily concentrated on the loss family, and most of its post-switch gains still come from loss, with little contribution from other families.

I CASE STUDIES: WHY THE MULTI-ROUND GAP EMERGES

This section uses four representative examples to explain why the multi-round gap above arises. Across all four cases, the same mechanism appears: after RL, GRPO, STEER, and DAPO increasingly revolve around the head family C4 (attention), because repeatedly emitting that family is the easiest way to align with the current reward-model preference or dataset prior. In the short term, this can preserve or even inflate hit-based scores. In the long term, it systematically removes non-head but genuinely high-yield families from the candidate set. SPOD does not help by enforcing a uniform distribution. It helps by suppressing this head-family bias, so that useful non-head families remain available before multi-round iteration even begins.

I.1 CASE 1: ALL THREE RL BASELINES COLLAPSE TO THE SAME FAMILY

This example is the cleanest illustration of semantic collapse after RL. The prompt comes from track 2 under the cross-track slice, at a relatively early state (27.7 minutes, $n_history = 3$). The ground-truth accepted direction is a C2 optimizer change: extending the learning-rate cooldown phase, with a gain of 1.805 minutes and a strong-tier label.

The key observation is that SFT does *not* hit the ground truth, but it still retains multiple families. After RL, however, all three RL baselines collapse completely to the head family C4. GRPO, STEER, and DAPO each produce 8/8 C4 candidates. At that point, the problem is no longer only ranking error: the correct family has disappeared from the candidate set entirely. Under the same RL setting, SPOD still preserves four families and includes a C2 candidate (Table 10). Its top-ranked candidate is still wrong, but the correct family remains alive in the pool.

Table 10: Case 1: the correct family disappears after RL. The ground truth is a C2 optimizer change, but all three RL baselines collapse to 8/8 C4 candidates.

Arm	Family distribution over 8 candidates	Hit@1
SPOD (β=1.0)	C1×3 / C2×1 / C3×2 / C4×2	miss
GRPO	C4×8	miss
STEER	C4×8	miss
DAPO	C4×8	miss
SFT	C1×4 / C3×2 / C4×2	miss

This case directly explains the long-horizon gap in the replay. A collapsed policy does not merely rank the wrong candidate first; it removes the correct family before ranking even starts. Once this happens repeatedly across prompts, non-head families such as C2 contribute less and less over time. SPOD's first gain is therefore not simply better top-1 ranking, but preventing RL from collapsing the candidate space so aggressively that the correct family disappears.

I.2 CASE 2: THE BASELINES KEEP BETTING ON ATTENTION WHILE SPOD PRESERVES THE OPTIMIZER DIRECTION

The second example shows that the same failure mode is not an isolated extreme case, but a stable search bias. This prompt also comes from track 2 under the cross-track slice (23.4 minutes, $n_history = 8$). The ground-truth accepted direction is again in C2: the Snoo optimizer, with gain 0.725.

SPOD produces a mixed candidate set and hits the ground truth at top-1, whereas GRPO, STEER, DAPO, and SFT all allocate five of their eight candidates to C4 and miss (Table 11). More importantly, this is not a matter of guessing the exact method name by chance. The top SPOD candidate stays within the optimizer / learning-rate family, while the baselines' top candidates continue to revolve around attention windows, attention skip connections, and qkv-related changes.

The interpretation is straightforward. These baselines have learned that staying near attention is the easiest way to keep reward high, even when the genuinely useful direction lies in the optimizer family. SPOD interrupts this bias. Its advantage is not that it forces an artificial uniform distribution, but that it keeps the optimizer family from disappearing under RL pressure.

Table 11: Case 2: SPOD preserves the optimizer family, while the baselines continue to bet on attention.

Arm	Family distribution	Hit@1	First line of top-scored candidate
SPOD (β=1.0)	C1×2 / C2×1 / C3×4 / C4×1	✓	`[C2] Faster step, reduced lr decay duration, reduced learning rate value range, improved parameter freeze & warmstart logic`
GRPO	C3×1 / C4×5 / C5×2	miss	`[C4] Attention window update, 1024x residual connection, no gate layer, 1024x qkv layer ...`
STEER	C3×1 / C4×5 / C5×2	miss	`[C3] Unwrapped Attention warmup with 15 min cooldown, skip connection no-warmup ...`
DAPO	C4×5 / C5×3	miss	`[C4] Attention window warmup, no attention skip connection for long sequences ...`
SFT	C3×1 / C4×5 / C5×2	miss	`[C4] Batched DGL attention`

I.3 Case 3: Betting on the Head Family Can Artificially Inflate Hit-Based Scores

The third example shows why hit-based metrics alone can overestimate collapsed policies. This prompt comes from track 1 under a late-stage slice (1.3 minutes, $n_history = 8$). The ground-truth accepted change contains both C1 and C4: an algebraic rewrite of XSA that preserves the same computation while improving execution speed.

Because the ground truth itself contains the head family C4, methods that collapse entirely to C4 can still score perfectly on hit-based metrics. In this example, GRPO, STEER, and DAPO all generate 8/8 C4 candidates and all achieve Hit@1; SFT and SPOD also hit (Table 12). The key point is not which method wins this example, but why the collapsed baselines can look strong at all: if the head family is common enough, repeatedly emitting it can directly raise Hit@1 and Hit@8, even when the candidate set has already degenerated semantically.

Table 12: Case 3: a collapsed policy can score well simply by betting on the head family.

Arm	Family distribution over 8 candidates	Hit@1
SPOD (β=1.0)	C1×4 / C3×1 / C4×3	✓
GRPO	C4×8	✓
STEER	C4×8	✓
DAPO	C4×8	✓
SFT	C1×1 / C3×1 / C4×5 / C5×1	✓

This is exactly the kind of behavior that can mask collapse if one only reports hit-based scores. The baselines are not demonstrating healthier search; they are benefiting from a strategy that keeps emitting the head family because that family is common and reward-friendly. This case is therefore direct evidence for why the main paper reports distributional alignment, coverage, and method-diversity metrics in addition to Hit@k.

I.4 Case 4: Even a Basic High-Gain Optimizer Change Is Missed by the Collapsed Baselines

The fourth example shows how damaging this bias can become in practice. This prompt comes from track 2 under the cross-track slice at an almost initial state (29.3 minutes, $n_history = 1$). The ground-truth accepted direction is a basic optimizer change in C2: adding standard weight decay, with gain 4.096 minutes, one of the largest gains in the batch and labeled strong-tier.

Such a basic, high-gain optimizer direction should be easy to recover. Yet the RL baselines systematically miss it. GRPO, STEER, and DAPO each allocate five of eight candidates to C4 and generate no C2 candidate at all. SFT still preserves one C2 candidate but does not rank it first. Only SPOD both retains C2 and hits the ground truth at top-1 (Table 13).

Table 13: Case 4: even a basic high-gain optimizer change is systematically missed by the collapsed baselines.

Arm	Family distribution	Hit@1	First line of top-scored candidate
SPOD (β=1.0)	C1×2 / C2×1 / C3×3 / C4×2	✓	`[C2] Single step SGD, increased step size and mini-batch size, decreased warmup steps ...`
GRPO	C1×1 / C3×2 / C4×5	miss	`[C4] Faster update attention weights, use long sequence embeddings, shorter training steps ...`
STEER	C1×1 / C3×2 / C4×5	miss	`[C3] Increase learning rate of RoPE layers by 10x and lower number of k-args for RoPE`
DAPO	C1×1 / C3×2 / C4×5	miss	`[C4] Single step attention`
SFT	C1×1 / C2×1 / C3×2 / C4×4	miss	`[C3] Merged batched attention with gradient attention`

This example is especially revealing because the prompt contains only one history item. The baselines therefore cannot plausibly be explained away as following a long, misleading attention-heavy context. Instead, they exhibit a learned default bias: once RL has reshaped the policy, generating attention-related candidates becomes the easiest general-purpose strategy, even when the prompt calls for a basic optimizer change. This is precisely why such methods underperform in the multi-round setting. If even an early, high-gain optimizer direction fails to survive candidate generation, repeated iteration will keep circling around the same head family and lose long-horizon reward.

I.5 Connection to the Multi-Round Results

Together, these four cases explain the multi-round results in the previous section. In the replay, GRPO-like methods can obtain some early reward from the head family, but once the high-yield directions in that family are consumed, they continue allocating mass back to the same family, effectively spending later rounds on homogeneous repetition. The single-prompt cases show the same mechanism in local form: the baselines do not merely rank slightly worse, but compress the candidate space into C4 so early that non-head yet high-yield families such as C2 often disappear altogether.

SPOD improves long-horizon performance because it changes this behavior at the source. It prevents RL from turning candidate generation into repeated head-family hacking, preserves the presence of useful non-head families in the pool, and therefore makes it possible for those families to keep contributing reward over repeated iteration.

J Three-Layer Training Dynamics of Diversity

This appendix asks a direct question. When reinforcement learning reduces the diversity of a policy, at which level does the loss actually happen? To answer it, we attach a lightweight diversity probe to the reward pipeline and record diversity throughout training. The probe only reads each generation batch. It does not touch the reward, the advantage, or the sampling state, so training itself is left unchanged. We separate diversity into three layers.

- **Token layer.** Token-level entropy of the policy output distribution.
- **Surface layer.** Lexical variation of the text, measured on the clean evaluation set by word-level Distinct-2, the TF-IDF Vendi score, and mean pairwise cosine distance (higher is more diverse); the training monitor additionally tracks Self-BLEU (lower is more diverse).

- **Method layer.** Variation of the idea that a response proposes. Every response is mapped to one of five idea categories (Sys/Kernel, Optimizer, Architecture, Attention, Loss). On these categories we track the category entropy, the top-1 category share, and the number of categories covered.

All seven methods use the same five-category scheme, so the method-layer axis is comparable across methods. Each run is monitored over 50 optimization steps.

The three layers behave very differently. Figure 13 plots the six signals over training. At the token layer, every method falls by 8 to 22 percent, with GRPO decreasing the least at 8.4 percent. This is ordinary entropy convergence and shows no diversity-relevant separation between methods. At the surface layer the methods narrow only mildly. SPOD keeps the most lexical variation, with Distinct-2 dropping about 2 percent against 3 to 8 percent for the baselines, and Self-BLEU stays low for all methods. The method layer is the watershed. SPOD is the only method whose category entropy stays flat and even rises, whose top-1 share stays the lowest, and whose category coverage stays the highest. Every baseline moves the other way. Category entropy falls, one category swells, and coverage shrinks.

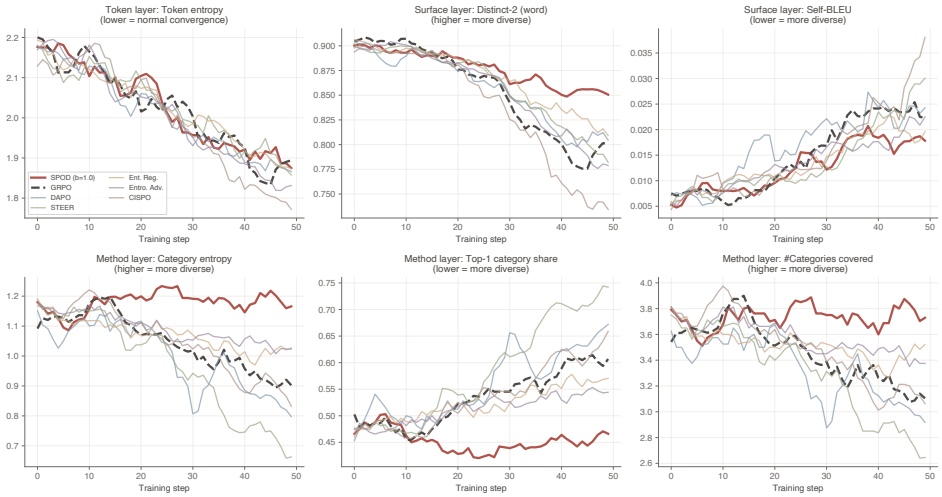

Figure 13: Six diversity signals over 50 training steps, grouped by layer. SPOD is the solid red curve and GRPO is the dashed black curve. Curves use a centered width-5 moving average. The token layer (top left) and the surface layer (top middle and top right) show little separation between methods. The method layer (bottom row) is where the methods split apart, and only SPOD stays diverse.

The collapse is a drift toward one idea. Figure 14 shows the idea-category mix as a stacked share over training. For every baseline one band grows steadily wider, most visibly the Attention category, while the other ideas fade. STEER and CISPO show the sharpest concentration. SPOD behaves in the opposite way. Its dominant band shrinks and probability mass flows into the under-used categories, so the mix becomes more balanced as training proceeds.

Quantitative summary. Table 14 reports the method-layer change on the clean evaluation set relative to the SFT baseline. SPOD is the only method that moves in the diverse direction on all three signals. Table 15 reports the token and surface layers.

Takeaway. Diversity collapse under reinforcement learning is not mainly a token or surface effect: at the token layer all methods converge similarly, and at the surface layer they narrow only slightly. The loss is concentrated at the method layer, where a policy keeps writing lexically varied text while quietly narrowing to a single kind of idea. Among the seven methods studied, only SPOD resists this method-layer collapse, and the probe shows it doing so during training rather than only at the final checkpoint.



Figure 14: Stacked idea-category share over training. A single band that keeps widening indicates collapse onto one idea. SPOD (top left) is the only method whose distribution stays balanced and in fact spreads toward the smaller categories.

Table 14: Method-layer change on the clean evaluation set relative to the SFT baseline. Arrows give the diverse direction. SPOD is the only method that improves on all three signals.

Method	Category entropy ↑	Top-1 share ↓	#Categories ↑
SPOD (β=1.0)	$+\mathbf{6.4}\%$	$-\mathbf{9.1}\%$	$+\mathbf{2.9}\%$
GRPO	-24.3%	$+29.1\%$	-18.4%
DAPO	-22.4%	$+27.3\%$	-16.5%
STEER	-21.1%	$+26.4\%$	-15.5%
Ent. Reg.	-26.6%	$+34.5\%$	-18.4%
Entro. Adv.	-9.4%	$+11.8\%$	-6.8%
CISPO	-28.6%	$+33.6\%$	-21.4%

Table 15: Token- and surface-layer change. Token entropy is each arm's training-log endpoint change; the surface metrics are clean-eval changes relative to SFT. SPOD narrows the least at the surface layer.

Method	Token entropy	Distinct-2 (word)	Vendi (TF-IDF)	Pairwise
SPOD (β=1.0)	-19.1%	$-\mathbf{2.0}\%$	$-\mathbf{1.8}\%$	$-\mathbf{3.9}\%$
GRPO	-8.4%	-3.5%	-2.3%	-4.6%
DAPO	-18.0%	-4.7%	-4.8%	-6.7%
STEER	-17.7%	-3.6%	-2.7%	-4.6%
Ent. Reg.	-15.3%	-5.6%	-3.8%	-5.7%
Entro. Adv.	-15.3%	-2.9%	$-\mathbf{1.8}\%$	-4.0%
CISPO	-21.9%	-7.5%	-5.7%	-7.7%

K Robustness to Method-Category Annotation

SPOD uses method-category labels to correct category-level reward bias, so a natural question is how sensitive it is to the quality of those labels. This question matters especially in the industrial setting, where no ground-truth taxonomy is available and the category signal must be inferred automatically. In this appendix, we test that sensitivity from four angles: weaker category extractors (Section K.1), noisy labels with full SPOD re-runs (Section K.2), vocabulary granularity (Section K.3), and perturbations of the ϕ table (Section K.4). Across all four tests, the same conclusion emerges. SPOD does not require a precise or fragile taxonomy. It only needs a category partition that is coarse but still informative.

K.1 Q1: Weaker category extractors

We first test whether SPOD still helps when the category signal is replaced by a much weaker one. This is important because, in practice, method categories may be extracted imperfectly rather than provided as clean labels. To stress-test this dependence, we construct two substantially weaker method-category extractors and use them in place of the category assignment used in the main experiments: (i) X_{cent}, a TF–IDF nearest-centroid classifier, and (ii) X_{kw}, a keyword-scoring rule-based extractor.

We then re-evaluate the same held-out outputs using category assignments produced by each weaker extractor. Because different extractors change the measurement scale, the absolute metric values are not the main point of comparison. The key question is whether SPOD still improves over GRPO in the same direction as in the main results. Table 16 shows that it does. For every metric that still provides a meaningful comparison, the direction of the SPOD-over-GRPO improvement is unchanged. The only exception is All-8-same-category, which becomes 0.000 for both methods under both weak extractors and therefore no longer distinguishes between them.

Table 16: Held-out metrics recomputed after replacing the category field with a weaker extractor. Entries are Δ = SPOD − GRPO. "Sign" counts how many of the two weaker extractors preserve the direction of the main result.

Metric	Dir.	Main exp. field	X_{cent}	X_{kw}	Sign
Quality					
Hit@1	↑	$+0.218$	$+0.130$	$+0.130$	2/2
Hit@8	↑	$+0.217$	$+0.087$	$+0.130$	2/2
Gold-category coverage	↑	$+0.305$	$+0.087$	$+0.207$	2/2
TV to library prior	↓	-0.375	-0.139	-0.070	2/2
Diversity					
distinct ideas	↑	$+0.786$	$+0.464$	$+0.250$	2/2
Top-1 category share	↓	-0.286	-0.170	-0.118	2/2
Mean max same-category	↓	-1.500	-0.679	-0.714	2/2
All-8-same-category	↓	-0.071	0.000	0.000	n/a
Total informative directions preserved					**14/14**

These results show that the main GRPO-to-SPOD improvement does not depend on a high-precision category extractor. Even after replacing the category signal with much weaker alternatives, the quality and diversity gains remain in the same direction.

K.2 Q2: Does SPOD remain useful when the method labels are noisy?

SPOD relies on method-category labels to apply category-aware reward correction. A natural concern, therefore, is whether this dependence makes it brittle: if the labels are imperfect, does SPOD fail quickly, or does it degrade gradually as the category signal becomes noisier?

To test this, we deliberately corrupt the method labels and then re-run SPOD using the corrupted annotations. Concretely, for each example, we keep the original label with probability $1 - \varepsilon$, and

with probability ε we replace it with a different category label. This lets us control how much category information remains available to SPOD.

What should happen under random label noise? SPOD's correction is category-dependent. If the labels are corrupted at random, the correction should not disappear all at once; instead, it should be progressively diluted because the category signal becomes less reliable. Under the simple noise model above, the expected correction for category c becomes

$$\phi_{\text{eff}}(c) = \left(1 - \frac{K}{K-1}\varepsilon\right)\phi(c),$$

where K is the number of categories. In other words, random label noise mainly scales down the original correction signal.

The correction vanishes when

$$\varepsilon^\star = \frac{K-1}{K}.$$

In our main setting, $K = 5$, so $\varepsilon^\star = 0.8$. This gives a clear prediction: as ε increases, SPOD should gradually move back toward the GRPO baseline. It should not collapse as soon as a modest amount of label noise is introduced.

Table 17(a) shows exactly this pattern. As the labels become noisier, SPOD weakens smoothly rather than failing abruptly. Across $\varepsilon \in [0, 0.6]$, it remains better than GRPO on all eight metrics. At $\varepsilon^\star = 0.8$, where the expected correction is zero, the results return to nearly the GRPO level. Overall, these results suggest that noisy labels mainly reduce the strength of SPOD's correction, rather than turning it into a harmful signal.

Table 17: Robustness of SPOD to noisy method labels and to category granularity, compared with a common GRPO reference row. Panel (a) corrupts the method labels and re-runs SPOD. Its *Scale* column reports how much of the original correction signal is expected to remain under that noise level. Panel (b) changes the number of method categories K. There, *Scale* reports the relative strength of the resulting correction signal, normalized so that the main-text setting ($K = 5$) equals 100%.

		Quality				Diversity			
Setting	Scale	Hit@1↑	Hit@8↑	Cov.↑	TV_{lib} ↓	#Ideas↑	Top-1↓	MaxSame↓	AllSame↓
GRPO	—	0.217	0.696	0.565	0.498	3.000	0.621	5.071	0.071
(a) Noisy method categories; scale $= \phi_{\text{eff}}$									
$\varepsilon = 0.0$	1.000	**0.435**	**0.913**	**0.870**	**0.123**	**3.786**	**0.335**	**3.571**	**0.000**
$\varepsilon = 0.1$	0.875	0.435	0.913	0.826	0.146	3.679	0.375	3.607	0.000
$\varepsilon = 0.2$	0.750	0.435	0.870	0.783	0.155	3.714	0.353	3.500	0.000
$\varepsilon = 0.3$	0.625	0.435	0.913	0.870	0.213	3.571	0.357	3.857	0.000
$\varepsilon = 0.4$	0.500	0.391	0.870	0.783	0.235	3.643	0.362	3.607	0.000
$\varepsilon = 0.5$	0.375	0.304	0.826	0.783	0.347	3.393	0.460	4.143	0.036
$\varepsilon = 0.6$	0.250	0.261	0.783	0.652	0.404	3.143	0.518	4.714	0.036
$\varepsilon^\star = 0.8$	0.000	0.261	0.696	0.565	0.489	3.036	0.607	5.036	0.071
$\varepsilon = 1.0$	-0.250	0.130	0.522	0.435	0.718	2.393	0.862	6.143	0.143
(b) Vocabulary granularity; row label $= K$									
1	0%	0.261	0.696	0.565	0.489	3.000	0.616	5.000	0.071
2	97%	0.435	0.870	0.870	0.143	3.750	0.384	3.786	0.000
3	108%	0.391	0.913	0.870	0.159	3.679	0.375	3.679	0.000
5	100%	**0.435**	**0.913**	**0.870**	**0.123**	**3.786**	**0.335**	**3.571**	**0.000**
8	88%	0.391	**0.913**	0.826	0.203	3.571	0.384	3.714	0.000
20	41%	0.391	0.826	0.696	0.363	3.357	0.451	4.286	0.036
40	21%	0.261	0.739	0.609	0.454	3.107	0.558	4.714	0.036
184	5%	0.217	0.696	0.609	0.492	2.964	0.629	5.107	0.071

K.3 Q3: VOCABULARY GRANULARITY

We next test whether SPOD only works at one carefully chosen category granularity. This matters because any practical taxonomy requires a design choice about how coarse or fine the category space should be. If SPOD depended on a single special vocabulary size, its usefulness would be much more limited.

The main text uses a five-category vocabulary. Table 17(b) shows that changing the number of categories mainly weakens SPOD toward GRPO rather than below it. At the coarsest extreme, $K = 1$, no category structure remains and SPOD moves back to the GRPO regime. At the finest extreme considered here, $K = 184$, the category space becomes so fine that the correction signal is spread too thinly, and the results again move back toward GRPO. The five-category setting used in the main text therefore does not behave like a fragile special case. It lies inside a fairly broad effective range, and nearby coarsening or refinement leaves the qualitative conclusion unchanged.

K.4 Q4: SENSITIVITY TO THE ϕ TABLE

Finally, we test whether SPOD depends on a finely calibrated correction table ϕ. Here, the ϕ table is the lookup table that assigns each category its correction score in the reward debiasing step. Even if the category assignments themselves are reasonable, the method could still be fragile if it required very precise numerical values for these scores.

Table 18 compares several informative variants of ϕ, including two independently calibrated tables, a sign-only version, and a sparse one-category version, against two controls: a constant ϕ and a random permutation. The goal is to see whether SPOD needs one very specific calibration, or whether it works as long as the table preserves the broad structure of category-level bias.

Table 18: Results under different ϕ tables. The first four rows are informative variants, and the last two are controls. Random permutation is averaged over 30 draws.

	Quality				Diversity			
ϕ table	Hit@1↑	Hit@8↑	Cov.↑	TV_{lib} ↓	#Ideas↑	Top-1↓	MaxSame↓	AllSame↓
GRPO (no correction)	0.217	0.696	0.565	0.498	3.000	0.621	5.071	0.071
Calibrated (main text)	**0.435**	**0.913**	**0.870**	**0.123**	**3.786**	**0.335**	**3.571**	**0.000**
Earlier calibration	0.391	0.870	0.783	0.238	3.643	0.433	3.964	0.000
$\mathrm{sign}(\phi)$ only	0.435	0.913	0.783	0.241	3.714	0.424	3.821	0.000
`C4` only ($\phi = 1$, else 0)	0.261	0.826	0.739	0.396	3.286	0.496	4.536	0.036
Control: $\phi \equiv 1$	0.217	0.826	0.696	0.431	3.357	0.558	4.607	0.036
Control: random perm.	0.217	0.696	0.609	0.512	3.036	0.649	4.964	0.071

Table 18 leads to a clear conclusion. SPOD does not depend on a highly specific calibration of the ϕ table. As long as the table preserves meaningful category-level structure, performance stays within a similar range. Once that structure is removed, as in the constant and permuted controls, the gains largely disappear. This shows that SPOD relies on the coarse pattern of category-level bias rather than on finely tuned numerical values.

Across Tables 16–18, the overall conclusion is consistent. SPOD does not require a precise or brittle taxonomy. Its gains persist under weaker category extractors, noisy method labels, changes in vocabulary granularity, and substantial perturbations of the ϕ table. As the category signal becomes less informative, SPOD degrades smoothly toward GRPO rather than failing abruptly or dropping below it.

L MULTI-SEED RESULTS AND STATISTICAL TESTS

We repeat the overall public-benchmark evaluation for all seven post-training methods across five random seeds $\{42, 211, 985, 43, 44\}$. Every run uses the same SFT initialization, reward model, training prompts, and 28-task test set, and differs only in the RL objective or the diversity-preserving

mechanism. Each task has eight generated candidates. Hit@1, Hit@8, and Coverage use the 23 tasks that carry a ground-truth method label, while the four diversity metrics use all 28 tasks. TV measures the deviation from the reference library category distribution. Arrows in the headers give the preferred direction, and bold marks the best method for each metric. Metric abbreviations follow Table 9.

The five per-seed tables below list the raw scores. On every seed SPOD attains the best value on seven of the eight metrics, including all four diversity metrics together with TV, Coverage, and Hit@8, and on three of the five seeds it is best on all eight.

Seed 42

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SPOD (β=1.0)	**0.435**	**0.913**	**0.870**	**0.123**	**3.786**	**0.335**	**3.571**	**0.000**
GRPO	0.217	0.696	0.565	0.498	3.000	0.621	5.071	0.071
DAPO	0.304	0.739	0.641	0.498	3.071	0.612	5.000	0.071
STEER	0.217	0.739	0.641	0.475	3.107	0.603	4.964	0.071
Ent. Reg.	0.217	0.783	0.652	0.524	3.000	0.643	5.286	0.071
Entro. Adv.	0.261	0.826	0.707	0.395	3.429	0.522	4.393	0.036
CISPO	0.217	0.696	0.565	0.515	2.893	0.643	5.250	0.071

Seed 211

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SPOD (β=1.0)	**0.478**	**0.783**	**0.728**	**0.181**	**3.679**	**0.335**	**3.571**	**0.000**
GRPO	0.261	0.696	0.565	0.484	3.036	0.612	5.000	0.071
DAPO	0.261	0.739	0.609	0.440	3.107	0.567	4.786	0.036
STEER	0.391	0.783	0.674	0.350	3.321	0.478	4.107	0.000
Ent. Reg.	0.304	0.652	0.522	0.569	2.750	0.692	5.607	0.071
Entro. Adv.	0.391	0.783	0.663	0.417	3.321	0.545	4.571	0.036
CISPO	0.261	0.696	0.565	0.542	2.929	0.661	5.393	0.071

Seed 985

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SPOD (β=1.0)	0.348	**0.870**	**0.826**	**0.114**	**3.750**	**0.357**	**3.679**	**0.000**
GRPO	**0.435**	0.739	0.609	0.480	3.036	0.598	4.929	0.071
DAPO	0.304	0.696	0.587	0.484	3.036	0.612	5.036	0.071
STEER	0.304	0.739	0.630	0.538	3.000	0.656	5.357	0.071
Ent. Reg.	0.174	0.739	0.620	0.524	3.107	0.638	5.214	0.071
Entro. Adv.	0.348	0.826	0.707	0.390	3.393	0.518	4.429	0.000
CISPO	0.217	0.783	0.663	0.480	3.071	0.607	4.964	0.071

Seed 43

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SPOD (β=1.0)	0.304	**0.870**	**0.815**	**0.118**	**3.786**	**0.371**	**3.750**	**0.000**
GRPO	**0.348**	0.739	0.609	0.444	3.143	0.571	4.750	0.036
DAPO	0.217	0.783	0.652	0.471	3.179	0.589	4.893	0.071
STEER	0.261	0.696	0.565	0.524	2.964	0.634	5.179	0.071
Ent. Reg.	0.304	0.783	0.685	0.475	3.107	0.603	4.964	0.071
Entro. Adv.	0.348	0.739	0.609	0.453	3.107	0.580	4.821	0.071
CISPO	0.304	0.739	0.630	0.515	3.000	0.643	5.250	0.071

Seed 44

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SPOD (β=1.0)	**0.435**	**0.913**	**0.870**	**0.109**	**3.786**	**0.379**	**3.750**	**0.000**
GRPO	0.217	0.739	0.609	0.511	3.036	0.638	5.214	0.071
DAPO	0.348	0.696	0.587	0.506	3.000	0.634	5.179	0.071
STEER	0.261	0.696	0.554	0.574	2.821	0.688	5.500	0.071
Ent. Reg.	0.217	0.739	0.630	0.471	3.179	0.580	4.857	0.036
Entro. Adv.	0.261	0.826	0.696	0.413	3.321	0.540	4.536	0.036
CISPO	0.261	0.783	0.663	0.507	3.179	0.621	5.107	0.071

Table 19 and Table 20 average these runs and report the sample standard deviation ($\mathrm{ddof} = 1$). SPOD ranks first by the mean on all eight metrics, and Entro. Adv. is the runner-up on all eight. The gap between the best and the second-best mean is large for the alignment and diversity metrics, for example $+0.146$ on Coverage, -0.285 on TV, and -0.886 on Max Within-Family Count.

Table 19: Public-benchmark quality and alignment: mean $\pm$ sample standard deviation over five seeds. Bold marks the best method.

Method	Hit@1↑	Hit@8↑	Cov.↑	TV↓
SPOD (β=1.0)	$\mathbf{0.400} \pm 0.071$	$\mathbf{0.870} \pm 0.053$	$\mathbf{0.822} \pm 0.058$	$\mathbf{0.129} \pm 0.029$
GRPO	0.296 ± 0.094	0.722 ± 0.024	0.591 ± 0.024	0.483 ± 0.025
DAPO	0.287 ± 0.050	0.730 ± 0.036	0.615 ± 0.030	0.480 ± 0.026
STEER	0.287 ± 0.066	0.730 ± 0.036	0.613 ± 0.051	0.492 ± 0.087
Ent. Reg.	0.243 ± 0.058	0.739 ± 0.053	0.622 ± 0.061	0.513 ± 0.041
Entro. Adv.	0.322 ± 0.058	0.800 ± 0.039	0.676 ± 0.042	0.414 ± 0.025
CISPO	0.252 ± 0.036	0.739 ± 0.043	0.617 ± 0.049	0.512 ± 0.022

Table 20: Public-benchmark diversity: mean $\pm$ sample standard deviation over five seeds. Metric abbreviations follow Table 9. Bold marks the best method.

Method	Dist.↑	TopFam↓	MaxInFam↓	Coll.↓
SPOD (β=1.0)	$\mathbf{3.757} \pm 0.047$	$\mathbf{0.355} \pm 0.020$	$\mathbf{3.664} \pm 0.090$	$\mathbf{0.000} \pm 0.000$
GRPO	3.050 ± 0.054	0.608 ± 0.025	4.993 ± 0.172	0.064 ± 0.016
DAPO	3.079 ± 0.069	0.603 ± 0.025	4.979 ± 0.149	0.064 ± 0.016
STEER	3.043 ± 0.186	0.612 ± 0.081	5.021 ± 0.549	0.057 ± 0.032
Ent. Reg.	3.029 ± 0.168	0.631 ± 0.043	5.186 ± 0.294	0.064 ± 0.016
Entro. Adv.	3.314 ± 0.125	0.541 ± 0.025	4.550 ± 0.169	0.036 ± 0.025
CISPO	3.014 ± 0.115	0.635 ± 0.021	5.193 ± 0.163	0.071 ± 0.000

We next test whether the advantage of SPOD over the runner-up is statistically reliable. The comparison uses two-sided paired permutation tests between SPOD and Entro. Adv. The paired unit is a (task, seed) pair, which gives $n = 140$ over the five seeds. For each metric the statistic is the observed difference between the two methods, oriented so that a positive value favors SPOD. Each of $B = 20{,}000$ permutations swaps the two methods within a matched unit and recomputes the statistic, with

$$p = \frac{1 + \sum_{b=1}^{B} \mathbf{1}\{|\Delta_b^*| \geq |\Delta_{\mathrm{obs}}|\}}{B + 1}.$$

TV and Top-Family Share are recomputed from pooled category counts rather than averaged over per-task differences. The test relies on exchangeability under the null and does not by itself model a population of independent training seeds. Holm and Benjamini–Hochberg (BH) corrections are applied across the eight metrics.

Table 21 collects the results. Coverage, TV, Distinct Method Count, Top-Family Share, and Max Within-Family Count are all significant after Holm and BH correction, with raw permutation p below 0.001. Hit@8 also clears the raw and BH criteria. The observed difference is positive for SPOD on every metric. The strongest corrected evidence therefore concerns distributional alignment and reduced category concentration, which are the properties SPOD is designed to improve.

Table 21: SPOD versus Entro. Adv. on the public benchmark: two-sided paired permutation tests with multiplicity-adjusted values. Positive Δ favors SPOD. A star marks adjusted $p < 0.05$.

Metric	Δ	z	p	Holm	BH
Hit@1	$+0.0783$	$+1.53$	.1750	.1750	.1750
Hit@8	$+0.0696$	$+2.54$	.0194	.0582	$.0259^{*}$
Cov.	$+0.1457$	$+4.48$	$< .0001$	$.0004^{*}$	$< .0001^{*}$
TV	$+0.2846$	$+9.55$	$< .0001$	$.0004^{*}$	$< .0001^{*}$
Dist.	$+0.4429$	$+5.33$	$< .0001$	$.0004^{*}$	$< .0001^{*}$
TopFam	$+0.1857$	$+5.96$	$< .0001$	$.0004^{*}$	$< .0001^{*}$
MaxInFam	$+0.8857$	$+6.35$	$< .0001$	$.0004^{*}$	$< .0001^{*}$
Coll.	$+0.0357$	$+2.22$	.0640	.1280	.0731

M LIMITATIONS

SPOD relies on method-category labels for category-aware reward correction. On benchmarks where such categories are available, they can be used directly; in new domains without an existing taxonomy, they must be obtained through semantic annotation. Appendix K shows that SPOD degrades gracefully rather than collapsing when the category signal is weaker, labels are noisy, or vocabulary size changes. Still, constructing and validating a useful taxonomy for a new repository domain requires some upfront effort. This is mainly an engineering cost rather than a fundamental limitation, since the annotation pipeline can be built largely through automatic LLM extraction with only light human checking.