DGCR: LEARNING FROM LOCAL CORRECTIONS FOR SUPER-TINY FUNCTION CALLING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable function calling makes super-tiny models promising for efficient tool-based automation. However, local errors can invalidate otherwise useful calls, while unsuccessful rollout groups may be discarded without contributing to learning. Dense rewards give partial credit for correct decisions but do not specify how to repair the remaining errors. We introduce Discarded-Group Counterfactual Repair (DGCR) to recover explicit corrective targets from these rollouts. For collector-rejected groups that satisfy a low-reward condition, DGCR uses training references to propose bounded structural edits and verifies their gains under ordinary and fixed call assignments with consistency checks. It then applies gain-weighted supervision to the repaired spans under student-generated prefixes, without assuming that the complete response is correct. On Qwen3-0.6B, DGCR outperforms strong prior methods on both BFCL v3 and ACEBench, suggesting that even discarded rollouts can provide useful learning signals for super-tiny function calling models.

1 INTRODUCTION

Function calling allows language models to interact with external tools through structured, executable requests. Equipping super-tiny models with this capability offers a practical route to tool-based automation in resource-constrained settings. Recent typed-decision systems such as Jev emphasize the value of fast, structured decisions that software can consume directly (TypeSafe AI, 2026). For autoregressive function calling models, this goal requires precise coordination of function selection, argument values, and output syntax. The Berkeley Function Calling Leaderboard (BFCL v3) and ACEBench evaluate these requirements across function-calling tasks (Patil et al., 2025; Chen et al., 2025), while format-following studies show that content quality alone does not ensure a valid structured response (Xia et al., 2024).

Super-tiny function calling models face two interrelated training challenges. (1) **Local errors can invalidate otherwise useful calls.** A response may select the right function and supply most arguments correctly, yet fail because of one incorrect value, a missing argument, or an extra field. Existing methods use dense rewards to distinguish such partial successes from completely incorrect responses. For example, STAR applies Similarity-guided Reinforcement Learning (Sim-RL) after Constrained Knowledge Distillation (CKD) and combines format checks with graded function-call similarity rewards (Ni et al., 2026). These scores give credit for correct decisions, but do not provide an explicit target for the remaining correction. (2) **Unsuccessful rollout groups can be discarded without contributing to learning.** Difficult prompts may yield no successful response within a finite group. Equal rewards produce zero group-relative advantages, and reward-based collectors may reject groups with insufficient scores or contrast. Filtering and resampling recover informative groups, while alternative advantage constructions seek learning signals from weak reward contrast (Ni et al., 2026; Yu et al., 2025; Le et al., 2025). Yet a rejected response can still admit a verifiable local improvement. Discarding it leaves both the correction and the already-generated training context unused.

These challenges suggest a shared opportunity: a response need not be completely correct to supply a useful local training target. We introduce Discarded-Group Counterfactual Repair (DGCR) to recover such targets from collector-rejected groups that satisfy a low-reward condition (Figure 1). DGCR identifies an explicit correction to a student response and supervises only the repaired span.

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.

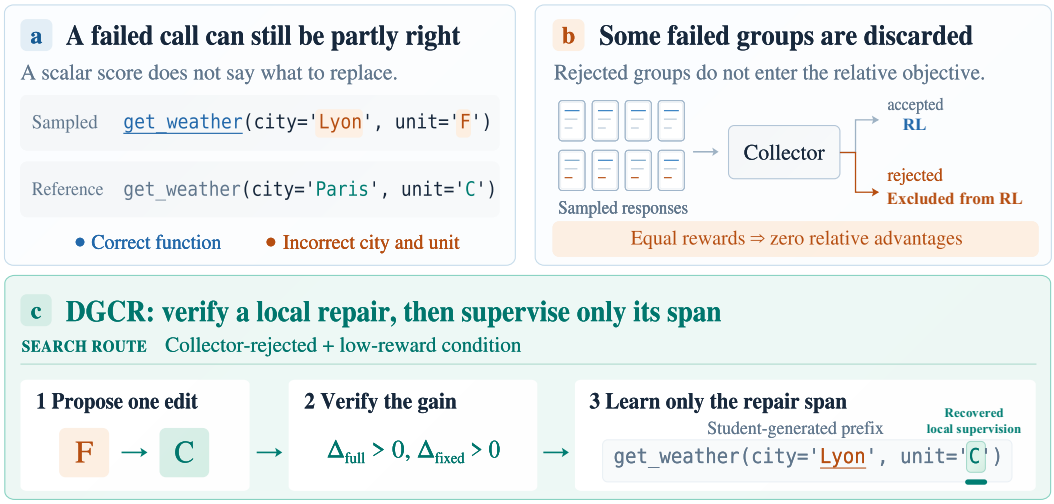


Figure 1: **Recovering verified local supervision from discarded rollout groups.** (a) A failed function call may contain correct decisions, but a scalar score does not specify the replacement tokens. (b) The collector excludes some groups from the relative objective; equal rewards illustrate a case with no within-group contrast. (c) For rejected groups satisfying the low-reward condition, DGCR proposes reference-supported local edits, verifies their gains under ordinary and controlled call assignments with consistency checks, and applies gain-weighted supervision only to the compiled repair span under the student-generated prefix. In this schematic example, replacing `F` with `C` supplies a local target even though `Lyon` remains incorrect.

Each verified correction supplies the target missing from scalar feedback and recovers supervision from an otherwise discarded rollout.

To construct this supervision, DGCR first uses training references to propose bounded edits to the abstract syntax trees (ASTs) of parseable student responses. It then verifies each edit under both ordinary and controlled assignments between predicted and reference calls, so a score increase caused by uncontrolled rematching alone cannot qualify as a local correction. Consistency checks also preserve the assignments of unedited calls. An accepted edit supplies a gain-weighted target for the repaired span under the student's own prefix. The resulting auxiliary objective teaches the local correction even when other errors remain, while accepted rollout groups continue to train through the relative objective.

Our main contributions can be summarized as follows.

- We propose Discarded-Group Counterfactual Repair (DGCR) to improve super-tiny function calling models by recovering explicit corrective supervision from eligible rollout groups discarded by the RL collector.
- We develop a repair procedure that combines reference-supported structural edits, gain verification under controlled call assignments, and gain-weighted span supervision. Each accepted repair provides a local target under the student's generated prefix without requiring the entire response to be correct.
- On Qwen3-0.6B, DGCR achieves an Overall Acc of 53.45% on BFCL v3 and a Summary score of 56.00% on ACEBench, compared with 51.76% and 53.10% for CKD+Sim-RL, the strongest evaluated baseline. Comparisons with reference supervision and random verified repairs further support its supervision construction and targeted allocation.

2 Related Work

Training Language Models for Function Calling. Recent work improves function calling through specialized training data, knowledge distillation, and reinforcement learning. ToolACE constructs

verified synthetic tool-use conversations, while xLAM develops action models through a scalable data and training pipeline (Liu et al., 2024; Zhang et al., 2024). For smaller models, STAR combines knowledge distillation with similarity-guided reinforcement learning (Sim-RL), which rewards partial correctness in structured calls (Ni et al., 2026). Such graded feedback allows training to distinguish responses that fail in different ways, even when none is fully correct. DGCR builds on this setting and uses the structured evaluator to verify explicit corrections to student responses. Whereas a similarity reward scores a response, an accepted DGCR repair also identifies target tokens for a local improvement. These targets provide auxiliary supervision for eligible rollouts discarded by the RL collector, while retaining the existing Sim-RL objective.

Recovering Learning Signals from Uninformative Rollout Groups. Group Relative Policy Optimization relies on reward differences among sampled responses, so groups with equal rewards are uninformative for its reward-relative update (Shao et al., 2024). Several methods address this issue by changing sampling or advantage construction. DAPO dynamically samples prompts, RL-ZVP shapes advantages for zero-variance prompts, and RC-GRPO conditions rollouts on reward tokens to influence their quality (Yu et al., 2025; Le et al., 2025; Zhong et al., 2026). Other approaches introduce additional supervision. SORT supplies reference supervision for all-wrong groups with token weights derived from plan conditioning, while ReCast introduces a positive anchor and a focused contrastive update for all-zero groups in generative recommendation (Le et al., 2026; Zhang et al., 2026). DGCR recovers supervision from an existing student response through bounded, reference-supported structural edits. Its search targets collector-rejected groups that satisfy a low-reward condition, an eligibility criterion distinct from exact zero reward variance. Each accepted edit must pass checks of its evaluator gain and call assignments before it supplies supervision restricted to the compiled repair span. This allows DGCR to learn from a verified partial improvement under the student's generated prefix without requiring the entire repaired response to be correct.

3 METHOD

DGCR augments the existing relative RL objective with local supervision recovered from eligible discarded rollouts (Figure 1). We first define the training setting, then describe how candidate edits become verified targets and enter actor updates.

3.1 PROBLEM SETUP

Let x contain a user request and tool definitions, y^* its reference call sequence, and p_θ the student policy. Our main setting starts from STAR's CKD policy and retains its Sim-RL reward (Ni et al., 2026). CKD combines top-k forward KL with selective tail regularization to initialize the student; Appendix B defines both inherited objectives. Training references are available to the evaluator and repair procedure, while the model generates from x alone.

For each prompt, the policy samples a group $g = (y_1, \ldots, y_G)$ of G responses with rewards $R_{g,i} = R_{\mathrm{Sim}}(y_i, y^*)$. The group-normalized advantage is

$$A_{g,i} = \frac{R_{g,i} - \bar{R}_g}{\sigma_g + \epsilon}, \qquad \bar{R}_g = \frac{1}{G}\sum_{j=1}^{G} R_{g,j}, \tag{1}$$

where σ_g is the reward standard deviation and $\epsilon > 0$ ensures a nonzero denominator. Equal rewards yield zero relative advantages, although a regularization term may still be present. The collector separately determines whether a group enters the RL update. We write $C_g = 1$ for an accepted group and $C_g = 0$ for a rejected one; this decision can depend on score levels or minimum contrast, so rejection and exact zero reward variance are distinct events.

DGCR preserves the relative objective on accepted groups and searches eligible rejected groups for a local correction. A candidate must pass gain and assignment checks and admit a valid token span before it supplies supervision. Accepted repairs enter a first-in, first-out (FIFO) buffer, from which actor updates consume target spans and their gain weights. We now specify this route and the edit neighborhood it searches.

3.2 RECOVERING TARGETS FROM REJECTED GROUPS

Repair construction begins with groups that satisfy $C_g = 0$ and $\max_i S_{g,i} \leq \tau_{\text{low}}$, where $S_{g,i}$ is the collector's routing score and τ_{low} is the threshold applied to that score. In Sim-RL, the routing score is the answer-similarity score in $[0, 1]$, with invalid outputs assigned zero; the total RL reward instead assigns invalid outputs -1. In the evaluated Sim-RL collector, the default low-score condition already implies rejection; the two predicates therefore identify a single search route. Exact zero reward variance is neither necessary nor sufficient for admission to this route.

For each parseable response y, the Sim-RL matcher visits predicted calls in output order and assigns each to the unmatched same-name reference with the highest argument similarity. Bounded search then constructs a finite set $\mathcal{E}(y)$ of candidate edits, each of which changes one AST unit against these references. The neighborhood includes argument-value replacement, missing-argument insertion, extra-argument deletion, and unambiguous function-name replacement for a single-call response. Replacement and insertion values come from exact reference values or finite admissible sets. These constraints keep candidate construction finite and tied to the training reference.

Each candidate $e \in \mathcal{E}(y)$ produces an edited response $y' = e(y)$ for verification. A usable candidate must be parseable, scorable under the required assignments, and representable by a nonempty token span. Across all parseable responses in a group, DGCR retains the candidate with the largest full evaluator gain among those that pass all checks. Deterministic tie-breaking by edit type, call index, and argument key yields at most one repair per group. The following checks determine which edits qualify for this selection.

3.3 VERIFICATION UNDER CONTROLLED CALL ASSIGNMENTS

A candidate can raise the evaluator score by changing its content or the call assignment. When several predictions compete for reference calls, editing one prediction may also change the references assigned to other calls. To establish whether the candidate supports a local target, we separate the score change under the original assignment from the contribution of rematching.

Suppress x and y^* and write the structured reward as $R(y) = F(y, M(y))$, where $M(y)$ is the predicted-to-reference assignment and F evaluates a response under a supplied assignment. For $M = M(y)$ and $M' = M(y')$, adding and subtracting $F(y', M)$ yields

$$R(y') - R(y) = \underbrace{F(y', M) - F(y, M)}_{\Delta_{\text{fixed}}(e)} + \underbrace{F(y', M') - F(y', M)}_{\Delta_{\text{assign}}(e)}. \tag{2}$$

The fixed-assignment term measures the change under the original correspondence, whereas the assignment term captures the net contribution from rematching the edited output. A positive full gain therefore need not imply a positive fixed-assignment gain, as the two-call example in Appendix C.2 illustrates.

For verification, let $M_e = M$ for argument edits. When both the predicted and reference sequences contain a single call, a function-name edit instead binds that call to the unique reference. This rebinding lets the corrected name receive credit when the original call was unmatched. The verifier compares the ordinary gain

$$\Delta_{\text{full}}(e) = R_{\text{Sim}}(y', y^*) - R_{\text{Sim}}(y, y^*) \tag{3}$$

with the controlled gain

$$\Delta_{\text{ctrl}}(e) = F(y', M_e) - F(y, M). \tag{4}$$

For argument edits, $\Delta_{\text{ctrl}} = \Delta_{\text{fixed}}$. Let $j(e)$ be the edited call index and $M_{-j(e)}$ denote unedited-call assignments, including unmatched status. Let $\text{resync}(y, y')$ and $L(e)$ denote the valid span boundary and target length defined in Section 3.4. For numerical tolerance $\eta_M \geq 0$, acceptance requires

$$\begin{aligned} \text{accept}(e) \iff\ & \Delta_{\text{full}}(e) > 0 \ \wedge\ \Delta_{\text{ctrl}}(e) > 0 \ \wedge\ |\Delta_{\text{full}}(e) - \Delta_{\text{ctrl}}(e)| \leq \eta_M \\ & \wedge\ M_{-j(e)} = M'_{-j(e)} \ \wedge\ \text{resync}(y, y') \neq \varnothing \ \wedge\ L(e) > 0. \end{aligned} \tag{5}$$

Positive controlled gain establishes improvement under the prescribed correspondence. Agreement with the ordinary gain excludes a net contribution from additional rematching. Preserving unedited-call assignments also rules out reassignments whose score changes cancel each other. This rule covers both argument edits and single-call function-name repairs. A valid, nonempty target span links the verified gain to the supervision below.

3.4 DENSE SUPERVISION ON THE REPAIRED SPAN

To turn an edit that passes the gain and assignment checks into a training target, the compiler serializes the edited response and tokenizes both versions. The shared prefix of the two token sequences defines t_0, and the repaired-side boundary t_1 delimits a nonempty span covering all changed tokens, ending where the unchanged suffix resumes or at a shared end-of-sequence token. We write $\mathrm{resync}(y, y') = (t_0, t_1)$ when these boundaries are valid and $\mathrm{resync}(y, y') = \varnothing$ otherwise. Candidates without a valid span are rejected, completing the acceptance test. The target thus preserves the student-generated prefix and covers the verified correction. A single structural edit may change several tokens, so token alignment determines the supervised span.

Let $r^* = y'_{[t_0,t_1)}$ be the target span and $L = L(e) = t_1 - t_0 > 0$ its length. The repair loss is

$$\ell_{\mathrm{repair}}(\theta; e) = -\frac{\omega(e)}{L} \sum_{\ell=1}^{L} \log p_\theta(r^*_\ell \mid x, y_{<t_0}, r^*_{<\ell}), \qquad \omega(e) = \min\left(\frac{\Delta_{\mathrm{full}}(e)}{\tau_R}, 1\right), \tag{6}$$

where $\tau_R > 0$ sets the gain scale. Length normalization gives each repair an average token loss, so longer serialized corrections do not receive more weight solely because they contain more tokens. The gain weight increases linearly with the verified improvement and saturates at one when $\Delta_{\mathrm{full}}(e) \geq \tau_R$. The loss mask is restricted to the correction and its serialization context between t_0 and t_1, so errors elsewhere receive no target labels from the auxiliary objective.

The first target token conditions on the prefix generated by the student, and subsequent target tokens condition on the repaired continuation through teacher forcing. This construction places supervision in a context the model has visited, as motivated by work on exposure bias and on-policy distillation (Ross et al., 2011; Bengio et al., 2015; Agarwal et al., 2024). A buffered prefix comes from the policy that generated its source rollout, while the repair loss is evaluated under the current policy. Each buffered example carries its target span and gain weight for integration with the RL objective.

3.5 TRAINING OBJECTIVE AND REPAIR SCHEDULING

Let H_g indicate that a rejected group produces an accepted repair, e_g denote its selected edit when $H_g = 1$, and $\lambda_{\mathrm{DGCR}} \geq 0$ control the repair-loss weight. Writing $\mathcal{L}^{(g)}_{\mathrm{RL}}$ for the Sim-RL contribution of an accepted group, the routed training objective is

$$\mathcal{L}_g = \begin{cases} \mathcal{L}^{(g)}_{\mathrm{RL}}, & C_g = 1, \\ \lambda_{\mathrm{DGCR}} \ell_{\mathrm{repair}}(\theta; e_g), & C_g = 0 \text{ and } H_g = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Equation (7) specifies the source of each training signal. Accepted repairs enter a FIFO buffer and are consumed in insertion order during actor updates. For the real repair examples $\mathcal{B}_R$ in an actor microbatch,

$$\mathcal{L}_{\mathrm{actor}} = \mathcal{L}_{\mathrm{RL}} + \lambda_{\mathrm{DGCR}} \mathcal{L}_{\mathrm{DGCR}}, \qquad \mathcal{L}_{\mathrm{DGCR}} = \begin{cases} \dfrac{1}{|\mathcal{B}_R|} \displaystyle\sum_{e \in \mathcal{B}_R} \ell_{\mathrm{repair}}(\theta; e), & |\mathcal{B}_R| > 0, \\ 0, & |\mathcal{B}_R| = 0. \end{cases} \tag{8}$$

The two objectives retain separate reductions, so the repair contribution is averaged over real repair examples. The repair quota limits the number of auxiliary examples relative to accepted RL responses, while λ_{DGCR} controls their loss coefficient. Buffered targets are distributed across actor microbatches up to this quota. Each update thus combines the relative objective on accepted groups with local supervision recovered from rejected groups. Appendix C.3 gives the quota and distributed-batch details.

3.6 WHY A REPAIR SUPPLIES A LEARNING SIGNAL

The auxiliary loss connects two distinct quantities: the improvement available through a structural edit and the learning signal supplied by its target tokens. To characterize the first, let

$\kappa(g) = \mathrm{Var}_{i \in g}[R_{\mathrm{Sim}}(y_i, y^*)]$ denote sampled reward contrast. For the finite edit set $\mathcal{E}(y)$ defined in Section 3.2, local improvability is

$$\rho(y) = \max(\{0\} \cup \{R_{\mathrm{Sim}}(e(y), y^*) - R_{\mathrm{Sim}}(y, y^*) \mid e \in \mathcal{E}(y)\}) . \tag{9}$$

A group can satisfy $\kappa(g) = 0$ while $\max_i \rho(y_i) > 0$. For example, all responses may share an incorrect unit that a bounded replacement improves. Positive ρ identifies a candidate improvement; the routing and verification conditions determine whether it yields a training target. Such a correction need not produce a completely correct response. We call an output *partially correct* when some function-calling decisions are correct despite errors elsewhere, and reserve *one-edit near miss* for an output that becomes completely correct after one admissible edit. DGCR can learn from the broader partially correct case, as illustrated in Figure 1.

Once an edit passes verification, its target span defines the conditional distribution fitted by the auxiliary objective. For a fixed prefix $u = (x, y_{<t_0})$, let $p_\theta^{(L)}$ denote the distribution over the next L tokens and δ_{r^*} the point-mass distribution on the verified repair. The span objective has the exact form

$$\ell_{\mathrm{repair}}(\theta; e) = \frac{\omega(e)}{L} D_{\mathrm{KL}}\left(\delta_{r^*} \,\middle\|\, p_\theta^{(L)}(\cdot \mid u)\right) . \tag{10}$$

At a repair position with next-token distribution q, target token $v = r_\ell^*$, and coefficient $c = \omega(e)/L$, the logit gradient is $c(q_j - \mathbf{1}_{\{j=v\}})$. The target-logit descent direction $c(1 - q_v)$ tends to c as $q_v \to 0$. An accepted repair therefore supplies an explicit local learning signal even when its source group contributes no reward-relative update. Appendix E derives the corresponding local gradient identities.

This gradient argument concerns the supervised context; shared parameters and joint optimization can also affect other predictions. The benchmark evaluation therefore measures whether these local updates improve the student's freely generated function calls.

Appendix F provides a complementary accounting of evaluator gains in repaired output distributions. That analysis separates the improvements constructed by search from their approximation by a learned policy; the implemented objective remains the local conditional loss in Equation (10).

4 Experiments

We test whether DGCR improves function calling, how supervision construction and allocation affect performance, and whether candidate verification improves the resulting targets. We then examine reward compatibility and model scale.

4.1 Experimental setup

Tasks and metrics. BFCL v3 evaluates curated Non-Live tasks, user-contributed Live queries, and interactive Multi-Turn tool use. ACEBench Normal tests argument types, single- and multi-turn calls, similar APIs, and user preferences (Patil et al., 2025; Chen et al., 2025). Both use their official checkers, including tool execution and state feedback for BFCL Multi-Turn. BFCL Overall Acc is the mean of its three partitions; ACEBench Summary averages the 12 Normal categories, rather than the five displayed groups. Appendix C.7 specifies the category weights and decoding protocol.

Comparisons and training. We follow STAR's training datasets (Ni et al., 2026); Appendix C.4 details the data sources and preparation. We retain the STAR baseline suite (Ni et al., 2026), including ToolRL (Qian et al., 2025) and LUFFY (Yan et al., 2025), with SFT denoting SFT-think and CKD+Sim-RL the closest comparison. CKD+Sim-RL and DGCR use the same CKD initialization and source RL corpus; DGCR adds repair supervision during RL. The shared RL settings are eight responses per prompt, rollout/training batch sizes of 128, a learning rate of 3×10^{-7}, temperature 1, and a 512-token generation cap. DGCR uses $\lambda_{\mathrm{DGCR}} = 1$, $\tau_{\mathrm{low}} = 0.01$, $\tau_R = 0.1$, and a repair quota of 20%. Appendix C.5 provides the remaining training settings.

Selection and uncertainty. Training budgets and hyperparameters were fixed before evaluation; every arm uses its final checkpoint at the prescribed budget. Neither checkpoints nor the default hyperparameters are selected by maximizing BFCL or ACEBench scores. The one-factor sweeps in Appendix D evaluate sensitivity around that default. The main tables report individual runs,

Table 1: **Results on BFCL v3.** All methods use Qwen3-0.6B; entries are individual-run point estimates. Bold and underlined values denote the best and second-best distinct scores in each column.

Method	Overall Acc (%)	Non-Live Acc (%)	Live Acc (%)	Multi-Turn Acc (%)
Standard methods				
Base-model	46.45	71.81	65.66	1.88
SFT	46.83	71.54	64.46	4.50
FKL	49.16	76.44	65.93	5.12
Recent methods				
ToolRL	46.04	64.81	66.55	6.75
LUFFY	48.94	76.75	64.59	5.48
GKD	46.16	67.62	67.61	3.25
STAR baselines				
CKD	49.44	76.56	66.15	5.62
Sim-RL	49.00	76.27	67.48	3.25
CKD+Sim-RL	<u>51.76</u>	<u>80.07</u>	<u>68.22</u>	<u>7.00</u>
DGCR	**53.45**	**82.15**	**70.54**	**7.67**

Table 2: **Results on ACEBench Normal.** All methods use Qwen3-0.6B; entries are individual-run point estimates. Bold and underlined values denote the best and second-best distinct scores in each column.

Method	Summary (%)	Atom (%)	Single-Turn (%)	Multi-Turn (%)	Similar API (%)	Preference (%)
Standard methods						
Base-model	27.20	37.70	19.50	10.00	36.00	6.00
SFT	28.70	42.30	14.00	9.00	34.00	10.00
FKL	36.80	52.30	16.00	16.00	42.00	**22.00**
Recent methods						
ToolRL	29.40	45.00	12.50	10.00	34.00	4.00
LUFFY	44.40	59.30	26.50	26.00	50.00	**22.00**
GKD	40.10	54.00	21.50	23.00	46.00	**22.00**
STAR baselines						
CKD	39.00	55.00	22.00	19.00	46.00	10.00
Sim-RL	39.30	55.30	23.50	21.00	52.00	10.00
CKD+Sim-RL	<u>53.10</u>	<u>69.70</u>	<u>35.50</u>	<u>33.00</u>	**62.00**	<u>20.00</u>
DGCR	**56.00**	**72.00**	**43.00**	**37.00**	<u>58.00</u>	**22.00**

while Appendix D.4 reports five-seed means and standard deviations for DGCR versus CKD+Sim-RL. These replications assess the overall comparison; the component comparisons in Table 3 use individual runs.

4.2 Function-calling performance

DGCR raises BFCL Overall Acc from 51.76% to 53.45% (Table 1). Non-Live and Live improve by 2.08% and 2.32%, respectively, whereas Multi-Turn rises from 7.00% to 7.67%. The strongest evidence is therefore for more accurate call construction in single-turn settings; the low Multi-Turn score leaves substantial difficulty in sustained tool interaction.

ACEBench Summary increases from 53.10% to 56.00% (Table 2). Single-Turn supplies the largest displayed gain, from 35.50% to 43.00%, and Multi-Turn rises from 33.00% to 37.00%. Atom and Preference also improve, but Similar API drops from 62.00% to 58.00%. Thus local correction improves aggregate accuracy without consistently improving discrimination between similar tools.

Table 3: **How should auxiliary supervision be constructed, allocated, and verified?** Individual-run scores. The supervision block compares reference targets and random verified repairs with a shared DGCR baseline. The routing block uses the same quota cap; the verifier block fixes unit weights and structural candidate order.

Repair configuration	BFCL Overall (%)	ACEBench Summary (%)
Supervision: how should targets be constructed and allocated?		
Targeted verified repair (DGCR)	**53.45**	**56.00**
Reference supervision	51.74	52.60
Random verified repair	52.39	54.30
Shared quota: which rollout groups enter repair search?		
Rejected low-reward groups only (DGCR)	**53.45**	**56.00**
All groups, including accepted or high-reward groups	39.56	45.30
Verification: which candidate edits become targets?		
Require positive gains and consistent call matching	**51.58**	**53.80**
Syntactic-only; omit gain and matching checks	46.18	49.20

ACEBench Normal Multi-Turn uses supplied dialogue histories; its improvement should not be equated with solving BFCL's interactive execution tasks.

The aggregate advantage also appears across five training seeds (Appendix D.4). Mean BFCL Overall Acc increases from $51.70 \pm 0.27\%$ to $53.41 \pm 0.42\%$, and mean ACEBench Summary from $53.12 \pm 0.68\%$ to $55.98 \pm 0.36\%$. The reported dispersions are standard deviations across runs.

4.3 How should auxiliary supervision be constructed and allocated?

Constructing supervision for the same prompts. DGCR outperforms reference supervision on both benchmarks (Table 3). For the same selected prompts, the reference control teacher-forces the training answer and applies cross-entropy (CE) at uniformly sampled reference positions. Each supervised token therefore conditions on the preceding reference tokens. DGCR instead supervises verified repair spans under student-generated prefixes. BFCL Overall Acc rises from 51.74% to 53.45%, and ACEBench Summary from 52.60% to 56.00%. These results support DGCR's joint construction of corrective targets and their conditioning contexts.

Allocating repair supervision. Targeted routing also outperforms random repair (Table 3). The random control selects verified repairs from the paired DGCR run's full rollout pool, with the same candidate generator, verifier, compiler, and local CE objective. Each selected repair retains its own gain weight, and each training arm uses its own RL rollouts. Relative to random repair, DGCR raises BFCL Overall Acc from 52.39% to 53.45% and ACEBench Summary from 54.30% to 56.00%. The result supports directing auxiliary supervision toward groups selected by the default route.

Verifying candidate improvements. Full verification improves BFCL Overall Acc from 46.18% to 51.58% and ACEBench Summary from 49.20% to 53.80% over syntactic-only selection (Table 3). Both arms retain the reference-guided generator, parsing, token round-trip, and local compiler, use unit weights, and select candidates in rollout-index, edit-kind, and call-index order. The controlled change is whether edits must pass the gain and call-assignment checks. Unit weights prevent gain weighting from suppressing nonpositive edits after those checks are removed. These results support the joint use of gain and assignment checks to select auxiliary targets.

Broader search under the same quota. Searching all rollout groups, including accepted or high-reward groups, lowers both benchmark scores relative to the default route (Table 3). This comparison retains full verification, gain weighting, the FIFO, and the 20% quota. It complements the random-repair experiment by testing which groups enter repair search before candidate selection.

Compatibility with response-level rewards. Holding CKD initialization and the structural repair evaluator fixed, DGCR improves both metrics with Binary, ToolRL, and Sim-RL rewards (Figure 2(a)).

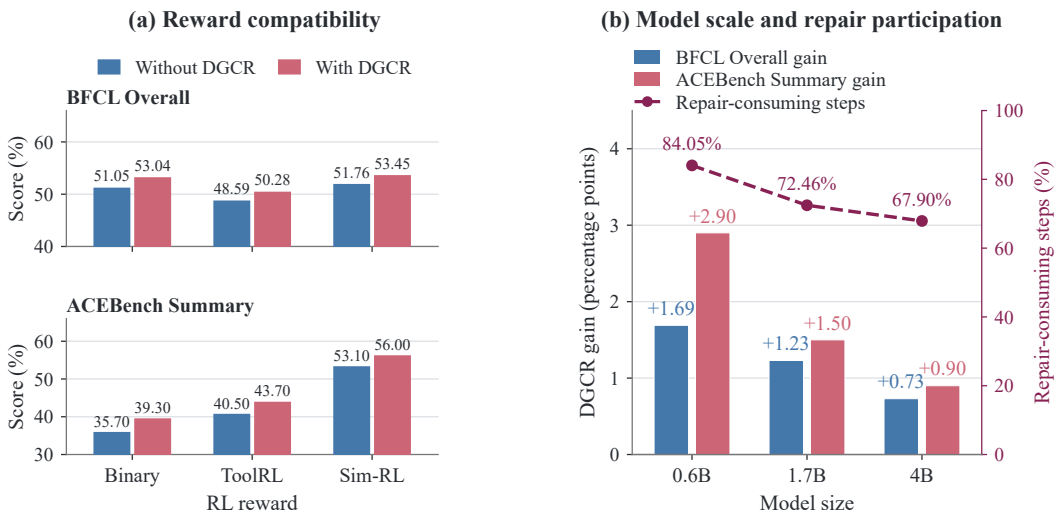


Figure 2: **Reward compatibility and scaling behavior.** (a) BFCL Overall and ACEBench Summary scores under three RL rewards from CKD initialization; blue and red bars denote training without and with DGCR, respectively. (b) Bars show DGCR's absolute score gains over STAR (%, left axis). The dashed line shows the fraction of *all* optimizer steps that consume at least one repair example (%, right axis). This is a step-level participation rate, not a response-error or group-repair rate. Absolute scale scores appear in Table 6.

The respective BFCL gains are 1.99%, 1.69%, and 1.69%; ACEBench gains are 3.60%, 3.20%, and 2.90%. The RL reward changes across these comparisons, while Sim-RL continues to score and verify repairs. These pairs test whether the same corrective supervision complements both sparse and graded RL feedback. Starting from SFT also improves BFCL and ACEBench by 1.42% and 3.40%, respectively (Appendix D.5).

4.4 Smaller models show higher repair participation and larger gains

Smaller models exhibit both higher participation in auxiliary supervision and larger benchmark gains (Figure 2(b)). A repair-consuming step uses at least one repair example, and the denominator includes all optimizer steps. This fraction decreases from 84.05% at 0.6B to 72.46% at 1.7B and 67.90% at 4B. It measures participation in the repair objective rather than the frequency of response errors or the fraction of rejected groups repaired.

Across the same sizes, BFCL gains narrow from 1.69% to 1.23% and 0.73%, while ACEBench gains decrease from 2.90% to 1.50% and 0.90%. The largest gains occur at 0.6B, consistent with the focus on super-tiny models, while positive gains at 4B show that the benefit extends beyond the smallest policy tested.

5 Conclusion

DGCR recovers explicit corrective targets from eligible rejected rollout groups through targeted routing, verified structural edits, and local supervision under student-generated prefixes. It improves function calling over CKD+Sim-RL, with higher mean scores across five training seeds. Reference-supervision and random-repair comparisons support the value of supervision construction and targeted allocation, while the verifier comparison supports checking candidate improvements before training on them. DGCR adds no language-model query for repair construction and no inference-time component.

AI USE STATEMENT

Generative AI tools assisted with language polishing and editing of the manuscript and with figure preparation. The authors are responsible for the technical content, experimental results, references, and the final manuscript.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Thomas Anthony, Zheng Tian, and David Barber. Thinking Fast and Slow with Deep Learning and Tree Search. *arXiv preprint arXiv:1705.08439*, 2017. URL https://arxiv.org/abs/1705.08439.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *arXiv preprint arXiv:1506.03099*, 2015. URL https://arxiv.org/abs/1506.03099.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, et al. ACEBench: Who Wins the Match Point in Tool Usage? *arXiv preprint arXiv:2501.12851*, 2025. URL https://arxiv.org/abs/2501.12851.

Fei Ding, Yongkang Zhang, Youwei Wang, and Zijian Zeng. Reducing Credit Assignment Variance via Counterfactual Reasoning Paths. *arXiv preprint arXiv:2605.16302*, 2026. URL https://arxiv.org/abs/2605.16302.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation. In *Conference on Empirical Methods in Natural Language Processing*, 2016. URL https://aclanthology.org/D16-1139/.

Duc Anh Le, Tien-Phat Nguyen, Thien Huu Nguyen, Linh Ngo Van, and Trung Le. Selective Off-Policy Reference Tuning with Plan Guidance. *arXiv preprint arXiv:2605.11505*, 2026. URL https://arxiv.org/abs/2605.11505.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No Prompt Left Behind: Exploiting Zero-Variance Prompts in LLM Reinforcement Learning via Entropy-Guided Advantage Shaping. *arXiv preprint arXiv:2509.21880*, 2025. URL https://arxiv.org/abs/2509.21880.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, et al. ToolACE: Winning the Points of LLM Function Calling. *arXiv preprint arXiv:2409.00920*, 2024. URL https://arxiv.org/abs/2409.00920.

Jiliang Ni, Jiachen Pu, Zhongyi Yang, Jingfeng Luo, and Conggang Hu. STAR: Similarity-guided Teacher-Assisted Refinement for Super-Tiny Function Calling Models. *arXiv preprint arXiv:2602.03022*, 2026. URL https://arxiv.org/abs/2602.03022.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *arXiv preprint arXiv:1910.00177*, 2019. URL https://arxiv.org/abs/1910.00177.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=eOLdGbXT6t.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics*, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

TypeSafe AI. Jev: Introduction. TypeSafe AI Documentation, 2026. URL https://docs.typesafe.ai/introduction. Accessed September 22, 2026.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. FoFo: A Benchmark to Evaluate LLMs' Format-Following Capability. *arXiv preprint arXiv:2402.18667*, 2024. URL https://arxiv.org/abs/2402.18667.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vO8LLoNWWk.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, et al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. *arXiv preprint arXiv:2409.03215*, 2024. URL https://arxiv.org/abs/2409.03215.

Peiyan Zhang, Hanmo Liu, Chengxuan Tong, Yuxia Wu, Wei Guo, and Yong Liu. ReCast: Recasting Learning Signals for Reinforcement Learning in Generative Recommendation. *arXiv preprint arXiv:2604.22169*, 2026. URL https://arxiv.org/abs/2604.22169.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. *arXiv preprint arXiv:2601.18734*, 2026. URL https://arxiv.org/abs/2601.18734.

Haitian Zhong, Jixiu Zhai, Lei Song, Jiang Bian, Qiang Liu, and Tieniu Tan. RC-GRPO: Reward-Conditioned Group Relative Policy Optimization for Multi-Turn Tool Calling Agents. *arXiv preprint arXiv:2602.03025*, 2026. URL https://arxiv.org/abs/2602.03025.

The appendices provide background, reproducibility details, additional experiments, and supporting analysis. Appendix A extends Section 2 with counterfactual credit assignment, learning in model-generated contexts, and search-based supervision. Appendix B defines CKD and Sim-RL, including their objectives and reward components. Appendix C specifies call matching, candidate construction, span compilation, loss reduction, training configuration, computational cost, prompts, evaluation metrics, the training algorithm, and limitations. Appendix D reports hyperparameter sensitivity, component-comparison interpretations, absolute scores across model sizes, five-seed results, and SFT initialization. Appendix E derives local probability and gradient identities and finite-sample discovery probabilities. Appendix F analyzes evaluator gains in repaired output distributions and the approximation needed to retain them in a learned policy.

A Extended Related Work

Counterfactual credit assignment. IBPO derives process-sensitive advantages from counterfactual trajectory comparisons (Ding et al., 2026). DGCR uses a bounded structural edit as the counterfactual and verifies its gain under ordinary and controlled call assignments. The accepted edit supplies a local supervised target under the student's prefix.

Learning in model-generated contexts. Sequence-level distillation trains a student to reproduce teacher sequences (Kim & Rush, 2016). DAgger, scheduled sampling, and on-policy distillation address training in contexts generated by the learner (Ross et al., 2011; Bengio et al., 2015; Agarwal et al., 2024; Gu et al., 2024). On-Policy Self-Distillation (OPSD) uses a solution-conditioned self-teacher to provide token distributions along student rollouts (Zhao et al., 2026). DGCR shares the use of student prefixes, but constructs point targets through reference-supported structural search and verification without a language-model teacher for repair construction.

Search and weighted supervised learning. The separation of structural search from parameter learning is related to Expert Iteration, while the gain-weighted likelihood objective is related to advantage-weighted regression (Anthony et al., 2017; Peng et al., 2019). In DGCR, the verified gain of a bounded correction determines the weight of supervision on its repaired span.

B Background on CKD and Sim-RL

STAR trains a compact function-calling model through Constrained Knowledge Distillation (CKD), followed by Similarity-guided Reinforcement Learning (Sim-RL) (Ni et al., 2026). We summarize these inherited components to specify the initialization and RL objective used by DGCR.

B.1 Constrained Knowledge Distillation

CKD matches the teacher's most probable tokens and selectively penalizes high-probability student tokens outside that set. Let h denote a token context along a teacher-generated trajectory, and let $p_T(v \mid h)$ and $p_\theta(v \mid h)$ be the teacher and student probabilities over the full vocabulary. Define $K_h = \mathrm{Top}_k(p_T(\cdot \mid h))$ and $B_h = \mathrm{Top}_m(p_\theta(\cdot \mid h)) \setminus K_h$. The objective is

$$\mathcal{L}_{\text{FKL-}k} = \mathbb{E}_{h\sim\mathcal{D}_T}\left[\sum_{v\in K_h} p_T(v \mid h)\log\frac{p_T(v \mid h)}{p_\theta(v \mid h)}\right], \tag{11}$$

$$\mathcal{L}_{\text{tail}} = \mathbb{E}_{h\sim\mathcal{D}_T}\left[\sum_{v\in B_h} p_\theta(v \mid h)\right], \tag{12}$$

$$\mathcal{L}_{\text{CKD}} = \mathcal{L}_{\text{FKL-}k} + \lambda_{\text{tail}}\mathcal{L}_{\text{tail}}. \tag{13}$$

Here $\mathcal{D}_T$ denotes the token contexts in the teacher-generated distillation data. The retained teacher probabilities are not renormalized within K_h. The tail term suppresses tokens favored by the student but excluded from the teacher's top-k set. STAR uses $k = m = 100$ and $\lambda_{\text{tail}} = 10$ in the inherited configuration. CKD supplies DGCR's starting policy; Appendix E gives its local gradient analysis.

B.2 SIMILARITY-GUIDED REINFORCEMENT LEARNING

Sim-RL refines the distilled policy with graded rewards for format compliance and answer similarity. It compares generated calls with training references without executing tools during reward computation.

Format and argument similarity. Let $F(y) \in \{0, 1\}$ indicate whether the response satisfies STAR's format checks. Under the Qwen template, these require one reasoning-tag pair, properly delimited JSON tool calls with function names and argument objects, available function names, and argument keys permitted by the tool schema. For calls c and c^* with the same function name, let $A(c)$ and $A(c^*)$ be their argument-key sets. Their similarity is

$$\mathrm{sim}(c, c^*) = \frac{\sum_{a \in A(c) \cap A(c^*)} s(c_a, c_a^*)}{|A(c) \cup A(c^*)|}, \tag{14}$$

with similarity one when both argument sets are empty. The value score s is ROUGE-L F1 for two strings, exact equality for numeric or Boolean values, and exact equality after string conversion otherwise.

Function-call and response rewards. Let P and Q be the predicted and reference call sequences, and let M be the one-to-one greedy assignment. Predictions are visited in output order and matched to the remaining same-name reference with the highest argument similarity. The function-call reward is

$$R_{\mathrm{fc}}(P, Q) = \frac{\sum_{(i,j) \in M} \mathrm{sim}(P_i, Q_j)}{|P| + |Q| - |M|}. \tag{15}$$

The denominator counts predicted calls and unmatched reference calls; the empty–empty case has score one. If the reference requires a text response, R_{response} is the ROUGE-L F1 score between the predicted and reference text, with score zero for an empty predicted response. Only the branch selected by the reference is active, and the other reward component is set to zero. STAR's composite reward is

$$R_{\mathrm{Sim}}(y, y^*) = (F(y) - 1) + F(y)\big(R_{\mathrm{fc}} + R_{\mathrm{response}}\big). \tag{16}$$

Invalid responses receive -1; valid responses receive an answer score in $[0, 1]$. The collector's routing score $S_{g,i}$ is the answer score and is zero for invalid outputs, whereas their total RL reward is -1.

Group-relative optimization. For G responses sampled from $p_{\theta_{\mathrm{old}}}(\cdot \mid x)$, Sim-RL assigns every token in response y_i the standardized advantage $A_{g,i}$ from Equation (1). Define the token likelihood ratio

$$r_{i,t}(\theta) = \frac{p_\theta(y_{i,t} \mid x, y_{i,<t})}{p_{\theta_{\mathrm{old}}}(y_{i,t} \mid x, y_{i,<t})}. \tag{17}$$

Writing $D_{i,t} = D_{\mathrm{KL}}(p_\theta(\cdot \mid x, y_{i,<t}) \| p_{\mathrm{ref}}(\cdot \mid x, y_{i,<t}))$, the GRPO objective has the form

$$\mathcal{J}_{\mathrm{SimRL}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\Big(\min\{r_{i,t}A_{g,i}, \mathrm{clip}(r_{i,t}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})A_{g,i}\} - \beta D_{i,t}\Big)\right]. \tag{18}$$

The expectation is over training prompt–reference pairs and sampled rollout groups; p_{ref} is the fixed RL reference policy, ϵ_{clip} is the clipping threshold, and β weights KL regularization. The training loss minimizes $-\mathcal{J}_{\mathrm{SimRL}}$. Equal rewards yield zero relative advantages, while the collector separately selects groups for the RL update. DGCR retains this objective on accepted groups and adds the repair loss in Equation (8). The evaluated implementation uses the KL approximation and reduction described in Table 4 and Appendix C.3.

C Implementation and Evaluation Details

C.1 Call matching and candidate construction

After grouping reference calls by function name, the Sim-RL matcher visits predicted calls in output order. Each prediction is assigned to the unmatched same-name reference with the highest argument-similarity score, with ties resolved by reference order. Predictions that receive no reference assignment retain unmatched status.

For matched calls, candidate enumeration considers shared keys with unequal values, missing keys, and extra keys. Both value replacement and missing-key insertion draw on exact reference values or finite admissible sets, while function-name replacement is considered only for a single-call output with an unambiguous target. Each candidate changes one AST unit. After verification, candidates are ranked by full gain, edit type, call index, and key. The scoring comparison retains the identity and order of all calls for the assignment-preservation check.

For argument edits, controlled evaluation retains the original reference assignment. A single-call function-name repair uses explicit rebinding to its unambiguous reference, because an unmatched original call cannot gain under a strictly frozen unmatched assignment. This case is recorded separately as single-call rebinding. In both cases, the verifier requires positive ordinary and controlled gains, agreement within the matching tolerance, and unchanged assignments for every unedited call.

C.2 An example of rematching gain

Consider an illustrative scorer that averages the scores of two matched calls. Both predictions share a function name, and matching proceeds greedily in prediction order with both references initially available. In the following matrices, rows index predictions and columns index references:

$$S_{\text{before}} = \begin{pmatrix} 0.6 & 0.5 \\ 0.9 & 0.0 \end{pmatrix}, \qquad S_{\text{after}} = \begin{pmatrix} 0.4 & 0.5 \\ 0.9 & 0.0 \end{pmatrix}. \tag{19}$$

Only the first prediction changes. Before the edit, greedy matching assigns it to reference 1 and the second prediction to reference 2, for a mean score of 0.3. After the edit, rematching assigns the first prediction to reference 2 and the second to reference 1, for a score of 0.7. Although the full gain is 0.4, the fixed-assignment score is 0.2 and its gain is -0.1, so the assignment term is 0.5. The negative fixed gain and the change to the second prediction's assignment are each sufficient for rejection by the verifier. This constructed example illustrates the rematching effect under greedy assignment.

C.3 Token span and loss reduction

The repair compiler uses the shared prefix and unchanged suffix of the original and edited token sequences to identify a nonempty span covering the complete edit, allowing a shared end-of-sequence token as the terminal boundary. It records the original prefix, repaired target tokens, active loss mask, and evaluator gain. Training applies teacher forcing within the target span under the original prefix. Tokens outside the active loss mask receive no positive target labels from the auxiliary objective.

Whereas the PPO path uses a masked mean over action tokens from accepted groups, the repair path first averages the weighted negative log-likelihood within each active span and then averages those losses across real repairs. The FIFO quota is the ceiling of the configured repair fraction times the PPO sample count in a training rollout batch; these examples are then distributed across microbatches. Any remaining repair slots are filled with zero-weight dummies for distributed shape compatibility. These dummies are excluded from the repair-example denominator, so a batch with no real repairs contributes zero.

C.4 Training data

STAR's public data preparation pipeline (Ni et al., 2026) combines four complementary sources. ToolMind supplies synthetic multi-step and multi-turn tool-use interactions from its graph-synthesized subset. xLAM provides validated function-calling examples, and ToolACE contributes diverse tool-use conversations in a cleaned Qwen-compatible format. HAMMER supplies irrelevant-query

examples in which the available tools do not satisfy the user's request. Data preparation resources are available in the STAR repository.[1]

The pipeline merges the formatted conversations and represents prompts, tool definitions, and reference responses with the Qwen chat template. Sim-RL uses the prompt–reference pairs, while CKD uses teacher-generated trajectories that include reasoning and final answers. DGCR retains this post-distillation training setting and uses training references to construct and verify corrective targets. BFCL v3 and ACEBench remain the evaluation benchmarks, as described in Appendix C.7.

C.5 Experimental configuration

Experiments use OpenRLHF on a server with four NVIDIA H200 GPUs. We follow STAR's training configuration for parameters shared with its SFT, CKD, and RL stages (Ni et al., 2026). Table 4 separates these inherited settings from the DGCR-specific ones. The closest baseline and DGCR share the CKD starting policy and RL corpus; the SFT-initialized comparison is described in Appendix D.5.

Table 4: **Training configuration.** Shared SFT, CKD, and RL settings follow STAR; DGCR parameters govern repair construction and learning.

Setting	Value
Hardware / framework	4 NVIDIA H200 GPUs / OpenRLHF
RL algorithm	GRPO with the STAR collector
RL learning rate	3×10^{-7}, constant
Rollout / training batch size	128 / 128
Responses per prompt	8
KL approximation / initial coefficient	k_2 / 10^{-3}
Rollout temperature / top-p	1.0 / 1.0
Maximum generated tokens	512
CKD learning rate / batch size	3×10^{-6} / 128
CKD tail coefficient / k / m	10 / 100 / 100
SFT learning rate / batch size	2×10^{-5} / 128
DGCR coefficient λ_{DGCR}	1
Low-reward threshold τ_{low}	0.01
Gain scale τ_R	0.1
Maximum repair fraction	0.2
Suffix matching window	4 tokens
Maximum repairs per group	1

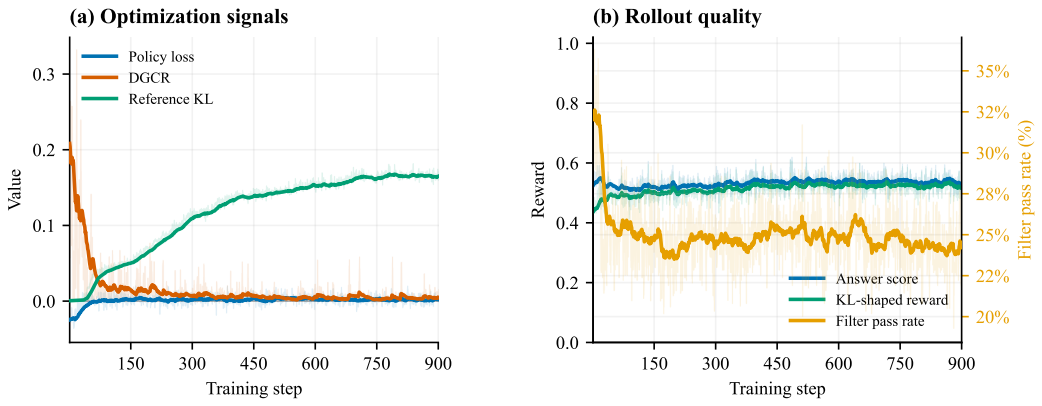


Figure 3: **DGCR training dynamics.**

[1] https://github.com/Qwen-Applications/STAR

Budget and selection rule. All training runs use one epoch and are evaluated at the final checkpoint, without selecting the best BFCL/ACEBench checkpoint. The default hyperparameters were also fixed before test evaluation. Dynamic filtering can change the number of rollouts required per update, so these model-quality comparisons do not establish end-to-end training efficiency under matched resources.

Computational cost. In a fixed-start timing run on one H200 GPU, repair construction averages 4.88 ms per searched group, with a median of 4.68 ms. The timing covers token round-trip checking, candidate enumeration, candidate scoring, verification, and span compilation, excluding rollout generation and optimizer updates.

C.6 Prompts

We follow STAR's prompt settings throughout training and evaluation (Ni et al., 2026). Training examples use the Qwen chat template, BFCL v3 uses the Qwen handler with STAR's system prompt, and ACEBench uses the official English prompt.[2]

C.7 Details of evaluation metrics

Both benchmarks evaluate generated function calls with their official checkers (Patil et al., 2025; Chen et al., 2025). We use the official BFCL v3 multi-turn interaction protocol and the official ACEBench Normal checker.

BFCL v3. The reported metrics capture complementary function-calling settings. Simple accuracy first averages Python, Java, and JavaScript equally. Non-Live then averages Simple, Multiple, Parallel, and Parallel-Multiple equally; Live uses the official sample-weighted average of its four AST categories. Multi-Turn averages Base, Missing Function, Missing Parameter, and Long Context equally. Overall is $(\mathrm{NonLive} + \mathrm{Live} + \mathrm{MultiTurn})/3$. Relevance and irrelevance tasks are excluded. Aggregation precedes rounding.

- **Overall Acc** summarizes performance across the Non-Live, Live, and Multi-Turn partitions.
- **Non-Live Acc** measures curated single-turn tasks, including simple, multiple-function, parallel-call, and combined multiple/parallel settings.
- **Live Acc** evaluates user-contributed queries and tool documentation designed to test generalization beyond static benchmark data.
- **Multi-Turn Acc** evaluates whether a model can preserve dialogue context, respond to follow-up requests, and complete tool interactions across several turns.

ACEBench Normal. We report the following metrics for the English Normal split. Atom averages six categories (Boolean, enum, list, number, deep object, and short object); Single-Turn and Multi-Turn each average two. Thus Summary is $(6\,\mathrm{Atom} + 2\,\mathrm{SingleTurn} + 2\,\mathrm{MultiTurn} + \mathrm{SimilarAPI} + \mathrm{Preference})/12$. Multi-Turn uses official end accuracy, rather than averaging intermediate-turn correctness. ACEBench decoding uses temperature 0, top-$p = 0.9$, and at most 1,024 new tokens; BFCL uses the official Qwen function-calling handler and the STAR system prompt. The evaluation configuration and checkpoint are held fixed within each paired comparison.

- **Summary** aggregates performance across the Normal categories into a single measure of tool-use capability.
- **Atom** evaluates argument construction across Boolean, enum, list, number, and object parameter types.
- **Single-Turn** measures function-call generation within one conversational turn.
- **Multi-Turn** measures context-sensitive tool use over dialogue histories that require a call to be revised or redirected.
- **Similar API** tests whether the model can distinguish among closely related tool descriptions.

[2] https://github.com/chenchen0103/ACEBench

- **Preference** tests whether API selection reflects contextual user preferences.

Comparison sources. The main tables retain the baseline suite and reported scores from STAR (Ni et al., 2026); the closest comparison uses the same initialization as DGCR. The ToolRL row in those tables represents a complete training baseline, whereas Figure 2(a) evaluates the ToolRL reward from CKD initialization. These comparisons address overall method performance and reward compatibility, respectively.

C.8 DGCR TRAINING ALGORITHM

Algorithm 1 summarizes how accepted rollout groups and verified repairs supply the two training terms. The buffer separates repair construction from consumption, so a repair can contribute during a later optimizer step. It retains the quota and separate loss normalization described in Appendix C.3.

Algorithm 1 Discarded-Group Counterfactual Repair

Require: Policy p_θ; training data $\mathcal{D}$; collector C; evaluator R_{Sim};
group size G; thresholds $\tau_{\mathrm{low}}, \tau_R$; coefficient λ_{DGCR}; quota fraction f
Ensure: Trained policy p_θ for ordinary function-call generation

$\mathcal{Q} \leftarrow \varnothing$ ▷ FIFO repair buffer
while training budget remains **do**
  $\mathcal{B}_{\mathrm{RL}} \leftarrow \varnothing$
  repeat
    Sample training pairs $(x, y^*) \sim \mathcal{D}$ and groups $g \sim p_\theta(\cdot \mid x)^G$
    Compute RL rewards, routing scores $S_{g,i}$, and collector decisions C_g
    for each sampled group g **do**
      if $C_g = 1$ **then**
        Append the responses in g to $\mathcal{B}_{\mathrm{RL}}$
      else if $\max_i S_{g,i} \le \tau_{\mathrm{low}}$ **then**
        $\mathcal{E}_g \leftarrow$ reference-supported single-unit AST edits of parseable responses
        Retain in $\mathcal{E}_g$ only edits that pass Equation (5) and token round-trip checks
        if $\mathcal{E}_g \neq \varnothing$ **then**
          $e^* \leftarrow \arg\max_{e \in \mathcal{E}_g} \Delta_{\mathrm{full}}(e)$ ▷ Deterministic ties; Appendix C.1
          $(u, r^*, \mu) \leftarrow$ original prefix, compiled target span, and loss mask of e^*
          $\omega \leftarrow \min(\Delta_{\mathrm{full}}(e^*)/\tau_R, 1)$
          Enqueue (u, r^*, μ, ω) in $\mathcal{Q}$
        end if
      end if
    end for
  until an accepted rollout batch is ready
  $N_{\mathrm{RL}} \leftarrow |\mathcal{B}_{\mathrm{RL}}|$; dequeue at most $\lceil f N_{\mathrm{RL}} \rceil$ repairs from $\mathcal{Q}$
  Distribute repairs across actor microbatches; pad unused slots with zero-weight dummies
  Reset accumulated actor gradients
  for each actor microbatch with accepted samples $\mathcal{B}_{\mathrm{RL}}^{(m)}$ and real repairs $\mathcal{B}_R^{(m)}$ **do**
    Compute $\mathcal{L}_{\mathrm{RL}}$ on $\mathcal{B}_{\mathrm{RL}}^{(m)}$ and $\mathcal{L}_{\mathrm{DGCR}}$ by Equation (8) ▷ Real repairs only; zero if empty
    Accumulate gradients of $\mathcal{L}_{\mathrm{RL}} + \lambda_{\mathrm{DGCR}} \mathcal{L}_{\mathrm{DGCR}}$
  end for
  Update θ with the accumulated gradients
end while
return p_θ

C.9 SCOPE AND LIMITATIONS

DGCR relies on parseable responses, training references, and edits that satisfy its routing, verification, and span conditions. It recovers supervision from a subset of rejected groups; coordinated errors and open-ended value synthesis may lie outside its bounded neighborhood. A verified evaluator gain does not guarantee complete correctness or successful downstream execution. Shared parameters can also change predictions outside the supervised span.

D Additional Ablations

D.1 Hyperparameter sensitivity

Table 5: **Hyperparameter sensitivity at 0.6B.** Each row changes one parameter from the default DGCR configuration.

Configuration	BFCL v3 Overall Acc (%)	ACEBench Summary (%)
Default DGCR	**53.45**	**56.00**
$\tau_{\text{low}} = 0.005$	50.92	53.50
$\tau_{\text{low}} = 0.02$	52.36	54.40
$\lambda_{\text{DGCR}} = 0.5$	50.74	53.00
$\lambda_{\text{DGCR}} = 2.0$	51.28	54.30
$\tau_R = 0.05$	52.81	54.60
$\tau_R = 0.2$	52.17	54.10

The three sweeps in Table 5 vary one parameter at a time around $\tau_{\text{low}} = 0.01$, $\lambda_{\text{DGCR}} = 1$, and $\tau_R = 0.1$. The default configuration performs best in the tested range. Moving the low-reward threshold in either direction reduces both metrics, and changing the repair coefficient produces a similar decline. Performance varies less with the gain scale within the tested interval, although the default still yields the highest scores on both benchmarks.

D.2 Interpretation of component comparisons

The reference-supervision comparison in Table 3 evaluates the target tokens and conditioning contexts together, while the random-repair comparison evaluates the default routing policy. The verifier ablation tests the gain and assignment checks jointly. The five-seed results in Appendix D.4 assess the overall DGCR versus CKD+Sim-RL comparison.

D.3 Absolute scores across model sizes

Table 6 provides the absolute scores underlying Figure 2(b). Repair-consuming steps have the same optimizer-step definition as in the figure.

Table 6: **Model scale, benchmark scores, and repair participation.** All entries are in %; Δ denotes absolute score gain. A repair-consuming step uses at least one real repair example.

	BFCL Overall			ACEBench Summary			Repair steps
Size	STAR	DGCR	Δ	STAR	DGCR	Δ	
0.6B	51.76	53.45	+1.69	53.10	56.00	+2.90	84.05
1.7B	56.05	57.28	+1.23	60.90	62.40	+1.50	72.46
4B	65.24	65.97	+0.73	74.10	75.00	+0.90	67.90

D.4 Across-seed aggregate results

We repeat the 0.6B DGCR and STAR (CKD+Sim-RL) comparison with four additional training seeds for five runs per method. Seed replication follows the same initialization, fixed-budget final-checkpoint rule, and prespecified hyperparameters as the corresponding comparison. Table 7 reports the mean and standard deviation of BFCL Overall Acc and ACEBench Summary across seeds. DGCR improves the mean scores over STAR by 1.71% and 2.86%, respectively. These replications support

the overall method comparison; they do not supply uncertainty estimates for the reference-supervision and random-repair controls.

Table 7: **Across-seed results at 0.6B.** Mean ± standard deviation over five training seeds (the original run and four additional seeds). Scores are percentages.

Method	BFCL v3 Overall Acc (%)	ACEBench Summary (%)
STAR (CKD+Sim-RL)	51.70 ± 0.27	53.12 ± 0.68
DGCR	$\mathbf{53.41 \pm 0.42}$	$\mathbf{55.98 \pm 0.36}$

D.5 SFT INITIALIZATION

Table 8: **SFT initialization at 0.6B.**

Method	BFCL v3 Overall Acc (%)	ACEBench Summary (%)
SFT	46.83	28.70
SFT w/ Sim-RL	50.41	38.90
SFT w/ Sim-RL + DGCR	**51.83**	**42.30**

Table 8 tests whether verified repair supervision remains useful without CKD initialization. Adding DGCR to SFT with Sim-RL improves BFCL v3 from 50.41% to 51.83% and ACEBench from 38.90% to 42.30%. The gain therefore extends to an SFT starting policy, while CKD-initialized DGCR retains the higher final scores of 53.45% and 56.00%.

E PROBABILITY REDISTRIBUTION AND LOCAL TARGET GRADIENTS

E.1 CKD OBJECTIVE AND LOCAL GRADIENT MODEL

At a fixed token context, let t_j and $q_j = \mathrm{softmax}(z)_j$ denote teacher and student probabilities. Let K be the teacher's top-k set, $B = \mathrm{Top}_m(q) \setminus K$ the penalized student set, $a = \sum_{j \in K} t_j$, and $b = \sum_{j \in B} q_j$. With penalty weight $\lambda_{\mathrm{tail}} > 0$, CKD uses

$$\mathcal{L}_{\mathrm{CKD}} = \sum_{j \in K} t_j \log \frac{t_j}{q_j} + \lambda_{\mathrm{tail}} b. \tag{20}$$

The retained teacher probabilities are not renormalized within the top-k set. Within a region of fixed set membership, the gradient derived in STAR, Appendix A.5 (Ni et al., 2026), is

$$\frac{\partial \mathcal{L}_{\mathrm{CKD}}}{\partial z_j} = a q_j - \mathbf{1}_{\{j \in K\}} t_j + \lambda_{\mathrm{tail}} q_j (\mathbf{1}_{\{j \in B\}} - b). \tag{21}$$

For non-top-k tokens outside B, the additional term relaxes the suppressive gradient relative to top-k forward KL at the same logits. The following identities characterize the incremental tail term under Euclidean gradient flow in logits. Teacher probabilities and the sets K, B remain fixed over a neighborhood without ranking ties, and s denotes flow time.

E.2 RELATIVE ODDS AND UTILITY

Consider the local tail-only gradient flow $\dot{z} = -\nabla_z(\lambda_{\mathrm{tail}} b)$, with s denoting flow time. For two unpenalized tokens $i, j \notin B$,

$$\frac{d}{ds} \log \frac{q_i}{q_j} = \lambda_{\mathrm{tail}} b (q_i - q_j). \tag{22}$$

The higher-probability token gains in relative odds. Although every unpenalized logit rises when $b > 0$, a low-probability token can lose probability mass as other logits increase faster. Redistribution thus depends on the current policy as well as the teacher-defined exclusion set.

The effect on a specified utility can be expressed by assigning a fixed one-decision value f_j to each token and writing $J = \sum_j q_j f_j$. Under the same tail-only logit flow,

$$\dot{J} = -\lambda_{\text{tail}} \sum_j q_j^2 (\mathbf{1}_{\{j \in B\}} - b)(f_j - J). \tag{23}$$

The sign depends jointly on the current probabilities, the penalized set, and the specified utility. The derivation below makes this dependence explicit and gives a constructed example in which an unpenalized useful token loses probability.

Since $q = \text{softmax}(z)$, differentiation of $b = \sum_{i \in B} q_i$ gives

$$\frac{\partial b}{\partial z_j} = q_j(\mathbf{1}_{\{j \in B\}} - b), \qquad \dot{z}_j = -\lambda_{\text{tail}} q_j (\mathbf{1}_{\{j \in B\}} - b). \tag{24}$$

For $i, j \notin B$, the identity $\log(q_i/q_j) = z_i - z_j$ immediately yields Equation (22). The identity describes the relative change between the two tokens.

For fixed utilities f_j, $J = \sum_j q_j f_j$ has gradient $\partial J/\partial z_j = q_j(f_j - J)$. Taking its inner product with $\dot{z}$ yields Equation (23). The calculation assumes that these token utilities have been specified externally.

For a fixed useful set U disjoint from B, put $v = \sum_{j \in U} q_j$, $S_2 = \sum_j q_j^2$, $S_{2,B} = \sum_{j \in B} q_j^2$, and $S_{2,U} = \sum_{j \in U} q_j^2$. Choosing $f_j = \mathbf{1}_{\{j \in U\}}$ gives

$$\dot{v} = \lambda_{\text{tail}} \left(b S_{2,U} + v S_{2,B} - b v S_2 \right). \tag{25}$$

For $b, v, \lambda_{\text{tail}} > 0$, useful mass increases exactly when $S_{2,U}/v + S_{2,B}/b > S_2$. This condition depends on the current mass allocation, even though all members of U are unpenalized.

Constructed counterexample. Take $q = (0.10, 0.01, 0.89)$, $B = \{1\}$, $U = \{2\}$, and $\lambda_{\text{tail}} = 1$. The tail-only logit velocities are $(-0.09, 0.001, 0.089)$. Their probability-weighted mean is 0.07022, so the useful second token has $\dot{q}_2 = 0.01(0.001 - 0.07022) = -0.0006922$. Its logit increases but its probability falls. The example illustrates how softmax normalization can separate logit growth from probability growth under the isolated tail term. The complete CKD objective additionally includes the teacher-directed forward-KL term.

E.3 A VERIFIED POSITIVE TARGET AT LOW PROBABILITY

For a target token v under a fixed repair prefix, let $c = \omega(e)/L > 0$ and consider the loss contribution $-c \log q_v$. Then

$$\frac{\partial[-c \log q_v]}{\partial z_j} = c(q_j - \mathbf{1}_{\{j=v\}}). \tag{26}$$

For any competitor $j \neq v$, the corresponding logit flow satisfies

$$\frac{d}{ds} \log \frac{q_v}{q_j} = c(1 - q_v + q_j) > 0. \tag{27}$$

In particular, the target-logit velocity tends to c as $q_v \to 0$, whereas the tail term's velocity for an unpenalized target is $\lambda_{\text{tail}} b q_v$ and tends to zero. These local directions distinguish selective suppression from supervision toward a verified target. In neural training, their effects also depend on the gain weight, span length, repair frequency, shared parameters, and joint objective.

E.4 FINITE-SAMPLE DISCOVERY OF LOCAL ALTERNATIVES

For a fixed student prefix $u = (x, y_{<t_0})$ and a particular repair span r^* of length L, define

$$s_\theta(u, r^*) = \prod_{\ell=1}^{L} p_\theta(r_\ell^* \mid u, r_{<\ell}^*). \tag{28}$$

Among G independent continuations from this prefix, the probability that at least one begins with the repair is

$$1 - (1 - s_\theta(u, r^*))^G \leq G s_\theta(u, r^*). \tag{29}$$

The equality follows from independent sampling, and the upper bound is the union bound. A low-probability correction can therefore be absent from a small group despite having positive probability under the policy. This calculation concerns continuation from a fixed prefix; complete-response success also depends on prefix visitation and the generated suffix.

For binary complete-response rewards with success probability p, independent sampling instead gives

$$\Pr(\kappa(g) = 0) = p^G + (1 - p)^G. \tag{30}$$

For $p = 0.01$ and $G = 8$, the zero-variance probability is approximately 92.27%. This illustrative binary-reward calculation is separate from the empirical rejection rate of Sim-RL, whose rewards are graded.

F Analysis of Repaired Output Distributions

This appendix quantifies the evaluator gains constructed by search and the distributional approximation required to retain those gains in a complete-output policy.

Proposition 1 (Score change after one repair per group)**.** *For a fixed prompt and reference, let p be the distribution of complete responses and sample a group $g \sim p^G$. A repair operator replaces at most one response with an accepted edit and otherwise leaves the group unchanged. Denote the accepted full-score gain by Δ_g, with $\Delta_g = 0$ for an unchanged group. If an index is selected uniformly from the repaired group and q denotes the marginal distribution of the selected response, then, for an integrable evaluator reward,*

$$\mathbb{E}_q[R_{\mathrm{Sim}}] - \mathbb{E}_p[R_{\mathrm{Sim}}] = \frac{1}{G}\mathbb{E}_g[\Delta_g]. \tag{31}$$

Proof. Write $\widetilde{g} = (\widetilde{y}_1, \ldots, \widetilde{y}_G)$ for the repaired group. Conditional on g and the chosen edit, uniform index selection gives

$$\frac{1}{G}\sum_{i=1}^{G} R_{\mathrm{Sim}}(\widetilde{y}_i) = \frac{1}{G}\sum_{i=1}^{G} R_{\mathrm{Sim}}(y_i) + \frac{\Delta_g}{G}. \tag{32}$$

Taking expectations yields Equation (31), since each original sample has marginal distribution p. □

For the routing in Section 3, the mean gain can be factored as

$$\mathbb{E}_g[\Delta_g] = \Pr(C_g = 0)\Pr(H_g = 1 \mid C_g = 0)\mathbb{E}[\Delta_g \mid C_g = 0, H_g = 1], \tag{33}$$

when the conditioning events have positive probability; otherwise, the expected gain is zero. The supply of improvements therefore depends on rejected-group frequency, repair acceptance, and accepted-edit gain. The candidate statistic ρ precedes verification, whereas H_g includes all acceptance conditions.

Corollary 2 (Approximation by a complete-output policy)**.** *If $R_{\mathrm{Sim}} \in [r_{\min}, r_{\max}]$ with $r_{\min} < r_{\max}$, then any distribution p' over complete outputs satisfies*

$$\mathbb{E}_{p'}[R_{\mathrm{Sim}}] - \mathbb{E}_p[R_{\mathrm{Sim}}] \geq \frac{\mathbb{E}_g[\Delta_g]}{G} - (r_{\max} - r_{\min})\,\mathrm{TV}(q, p'). \tag{34}$$

If $D_{\mathrm{KL}}(q\|p')$ is finite, the right-hand side is at least

$$\frac{\mathbb{E}_g[\Delta_g]}{G} - (r_{\max} - r_{\min})\sqrt{\frac{1}{2}D_{\mathrm{KL}}(q\|p')}. \tag{35}$$

Proof. Normalize the reward to $[0, 1]$ by subtracting $r_{\min}$ and dividing by $r_{\max} - r_{\min}$. Its expectation difference is bounded by $\mathrm{TV}(q, p')$, so the original reward difference is bounded by $(r_{\max} - r_{\min})\,\mathrm{TV}(q, p')$. Combining this bound with Equation (31) gives the first inequality, and Pinsker's inequality gives the second. □

STAR's composite reward lies in $[-1, 1]$, so its reward-range factor is two; normalization to $[0, 1]$ would give a factor of one. The corollary concerns complete-output distributions, whereas the implemented span loss optimizes the conditional distribution in Equation (10). Accordingly, the identity quantifies constructed improvements, and task evaluation measures their realization by the learned policy.