LOCALIZED GRPO: UTILIZING DENSE REWARD SIGNAL FROM LLM-JUDGE ERROR LOCALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning for LLMs is shifting from PPO toward *critic-free*, group-relative methods such as GRPO, which remove the value network that PPO uses to form token-level advantages. Vanilla GRPO instead forms one group-relative advantage per rollout and broadcasts it to every token, so the algorithm has no explicit mechanism for assigning reward credit within a response. On verifiable reasoning tasks, recent methods address this limitation using *intrinsic* proxies such as token entropy, likelihoods, or Monte-Carlo rollout estimates. We study a complementary setting: generation tasks where external LLM judges can identify localized quality signals in a response, such as error spans in machine translation (MT) and supported or hallucinated spans in long-form factuality. We convert these explicit fine-grained span judgments into per-token rewards, and introduce L-GRPO, a localized GRPO advantage estimator that uses those rewards while preserving GRPO's sequence-level group-relative signal. On document-level MT across ten language pairs, L-GRPO reduces LLM-judged error density by 7.2% relative to GRPO. On long-form factuality, it improves factual precision across four benchmarks by 9.6% relative. L-GRPO also outperforms two strong baselines on both tasks: PPO, which uses a learned critic, and SDPO, which adds feedback-conditioned self-distillation.

1 INTRODUCTION

Reinforcement learning has become a central component of LLM post-training (Ouyang et al., 2022). Early RLHF systems commonly used PPO (Schulman et al., 2017), whose learned value function can turn rewards observed at different positions into token-level advantages. However, recent group-relative methods such as GRPO (Shao et al., 2024) remove the value model training to avoid its computational cost and instability. They instead compare a group of responses sampled for the same prompt. Standard GRPO reduces each response to a scalar reward, normalizes these rewards within the group, and broadcasts the resulting advantage to every generated token of the response. Consequently, vanilla GRPO has no mechanism for utilizing dense per-token rewards within a response, a limitation that becomes more severe for long outputs where a small number of localized errors can determine overall quality.

A growing line of work addresses this limitation within GRPO by estimating finer-grained advantages from intrinsic signals such as token policy entropy (Tan et al., 2025). Self-Distillation Policy Optimization (SDPO) similarly derives dense token-level training signals from disagreement between the policy and a feedback-conditioned self-teacher (Hübotter et al., 2026). These model-derived signals provide useful attribution in reinforcement learning with verifiable rewards (RLVR), where the environment supplies only an overall correctness reward, but the attribution remains indirect: a high-entropy or low-likelihood token, or one whose probability changes substantially under the self-teacher, is not necessarily erroneous. For a broad class of generation tasks, however, evaluators can directly provide dense per-token rewards. This includes MT, factuality and grounding (Niu et al., 2024), and detoxification (Wu et al., 2023; Cao et al., 2024). In these settings, LLM judges can provide explicit fine-grained supervision by marking localized errors. Their locations therefore provide a more direct attribution signal than inferring one from intrinsic model statistics.

Prior work has demonstrated the value of such localized rewards in PPO training, where a learned value network turns them into token-level advantages. In this work, we address the critic-free ana-

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.

logue: how to use explicit span-level feedback in GRPO, where no value network exists. To this end, we introduce Localized GRPO (L-GRPO). L-GRPO maps LLM judge verdicts to token rewards, which are then used to augment the standard GRPO advantage with a token-level redistribution term:

$$\hat{A}_{i,t}^{\mathrm{LGRPO}} = \underbrace{\hat{A}_{i}^{\mathrm{GRPO}}}_{\text{group-relative broadcast}} + \alpha \underbrace{\hat{A}_{i,t}^{\mathrm{loc}}}_{\text{localized redistribution}} \tag{1}$$

The GRPO term retains the response-level comparison, while the localized term adjusts the advantages assigned to specific spans rewarded or penalized by the judge, without training a value model.

We evaluate L-GRPO on two established long-form generation tasks: document-level MT and long-form factuality, i.e., producing factually correct and verifiable responses to general information-seeking prompts. We show the effectiveness of L-GRPO through consistent, statistically significant gains over GRPO when fine-tuning Qwen3-8B (Yang et al., 2025). Our contributions are fourfold:

(i) We introduce L-GRPO, a critic-free advantage estimator that augments GRPO's group-relative broadcast with a mean-zero token-level redistribution term derived from LLM-judge error locations; it provably preserves each rollout's mean advantage and requires neither a value network (unlike PPO-based fine-grained-reward work) nor extra prefix-conditioned rollouts (unlike SPO (Guo et al., 2025)).

(ii) We show analytically and empirically why the rollout-level term must be retained: a naive token-only estimator has advantages that vanish as $\sqrt{p}$ for unscored tokens and diverge as $p^{-1/2}$ for scored ones under sparse span rewards ($p \to 0$), and collapses in training on both tasks.

(iii) We provide concrete recipes for converting span-level judge feedback into per-token rewards, including a matching heuristic that localizes 98% of spans, character-length-scaled MQM weights that prevent omission reward hacking in MT, and a refined VeriScore pipeline with span-localized four-way claim verdicts for factuality.

(iv) Under controlled comparisons on Qwen3-8B, L-GRPO reduces MT error density by 7.2% across ten language pairs and improves factual precision by 9.6% relative across four benchmarks, outperforming GRPO, PPO, and SDPO, while reaching GRPO's final precision with 34.3% fewer rollouts on average.

2 Related Work

Fine-grained rewards with PPO. There is extensive prior work optimizing token-, span-, or segment-level rewards using PPO. Wu et al. (2023) and Yoon et al. (2024) learn granular reward models in Fine-Grained RLHF and TLCR; Cao et al. (2024) and Wang et al. (2026a) derive rewards from LLM-generated critiques in RELC and Text2Grad; and Chan et al. (2024) redistribute a scalar reward using reward-model attention. These methods optimize the localized rewards primarily with PPO. This requires training a value model alongside the policy, and errors in its reward-to-go predictions directly affect the resulting advantages. L-GRPO consistently outperforms a PPO baseline using the same policy initialization, data, and localized rewards (see §5).

Credit assignment without a learned value model. Critic-free methods avoid value-model training, but must obtain finer-grained credit through other signals. SPO (Guo et al., 2025) estimates segment advantages by comparing rewards from rollouts conditioned on two prefixes: one ending before the segment and one including it. This avoids training a critic but requires substantially more rollout sampling than GRPO. GTPO (Tan et al., 2025) uses policy entropy to redistribute a response-level reward across tokens: high-entropy tokens in successful responses receive larger rewards, while low-entropy tokens in unsuccessful responses receive larger penalties. GTPO is useful in RLVR settings with only outcome rewards, but it cannot use the fine-grained LLM-judge feedback available in the generation tasks we study. By contrast, SDPO (Hübotter et al., 2026) converts such feedback into token-level training signals by conditioning a self-teacher on the judge's feedback and measuring the resulting changes in next-token distributions. However, the induced probability changes need not coincide with the locations of the errors identified by the judge. We show that L-GRPO, which uses those locations themselves as its attribution signal, outperforms SDPO on both tasks.

Machine translation. In MT, LLM-judge based evaluation has become the dominant paradigm. Methods such as GEMBA-MQM (Kocmi & Federmann, 2023), AutoMQM (Fernandes et al., 2023), EAPrompt (Lu et al., 2023), MQM-APE (Lu et al., 2025), and RUBRIC-MQM (Kim, 2025) prompt LLM judges to annotate MQM-style error spans. The latest WMT25 metrics shared task (Lavie et al., 2025) reported that LLM judges reliably rank translation systems and assess the overall quality of individual translations, but still underperform human annotators at identifying the exact error spans and their severities. We show that LLM-judged error spans, even though noisy, are a useful RL training signal: when converted into per-token rewards and optimized with L-GRPO (§3), they improve translation quality on test documents over GRPO. Closest to our work is Ramos et al. (2025), who use PPO to optimize localized XCOMET (Guerreiro et al., 2024) rewards for MT; we instead study localized rewards under GRPO.

Long-form factuality. Auto-graders such as FActScore (Min et al., 2023), SAFE (Wei et al., 2024), and VeriScore (Song et al., 2024) evaluate long-form factuality using an LLM-judged pipeline that first extracts independent atomic claims from the response, retrieves evidence for each claim from a web search engine and assigns factual or not-factual verdicts to each claim based on the retrieved evidence . Closest to our work is Chen et al. (2025b), who improve long-form factuality by using VeriScore-derived rewards with GRPO. However, their approach aggregates the claim verdicts into one scalar reward. TruthRL (Wei et al., 2026) likewise uses GRPO with a scalar ternary reward that distinguishes correct answers, hallucinations, and abstentions. L-GRPO does not collapse the VeriScore rewards into a scalar and takes advantage of the locations of the factuality verdicts to improve upon the above approches.

3 METHOD

3.1 PRELIMINARIES AND NOTATIONS

For a prompt x_i, the policy π_θ generates a rollout $y_i = (y_{i,1}, \ldots, y_{i,T_i})$ where $y_{i,t}$ is a token and T_i is the rollout length. The state before generating token $y_{i,t}$ is defined as $s_{i,t} = (x_i, y_{i,<t})$, consisting of the prompt and the first $t-1$ generated tokens. Let $r_{i,t}$ denote the reward assigned to token $y_{i,t}$ and $\hat{A}_{i,t}$ denote its advantage. Policy-gradient methods update π_θ by scaling each token's log-probability gradient $\nabla_\theta \log \pi_\theta(y_{i,t} \mid s_{i,t})$ by its advantage $\hat{A}_{i,t}$.[1] A key design choice is therefore the mapping from token rewards $\{r_{i,t}\}$ to token advantages $\{\hat{A}_{i,t}\}$. PPO converts token rewards into token-level advantages using a value network V_ψ trained alongside the policy. For the token-prefix state $s_{i,t}$, the value network is trained using a regression loss to predict the expected discounted return over the remainder of the rollout. Given the value estimates, PPO computes token-level advantages with generalized advantage estimation (GAE) (Schulman et al., 2016):

$$\hat{A}_{i,t} = \sum_{l=t}^{T_i} (\gamma\lambda)^{l-t}\, \delta_{i,l}, \quad \delta_{i,l} = r_{i,l} + \gamma V_\psi(s_{i,l+1}) - V_\psi(s_{i,l}). \tag{2}$$

$\delta_{i,l}$ is the temporal-difference residual at token l, $\gamma \in [0,1]$ is the reward discount factor and $\lambda \in [0,1]$ controls how rapidly contributions from future residuals decay.

GRPO, by contrast, removes the value network and therefore has no learned mechanism for converting token-level rewards into token-level advantages. In its standard form, for each prompt x, it samples a group of G rollouts $\{y_1, \ldots, y_G\}$ and reduces each rollout's token rewards to the scalar return $S_i = \sum_{t=1}^{T_i} r_{i,t}$. From these returns, GRPO forms a group-normalized advantage that it *broadcasts* to every token:

$$\hat{A}^{\text{GRPO}}_{i,t} = \hat{A}^{\text{GRPO}}_{i} = \frac{S_i - \mu_S}{\sigma_S + \epsilon}, \tag{3}$$

with μ_S, σ_S the mean and standard deviation of the returns $\{S_j\}$ over the group. Thus, GRPO uses token rewards only through their sum; their positions within the rollout have no effect on the update.

[1] PPO and GRPO additionally apply policy-ratio weighting, clipping, and KL regularization, which we omit here since our focus is the advantage estimator rather than these standard stabilization terms.

Self-Distillation Policy Optimization. SDPO converts rich textual feedback from an LLM judge into a dense training signal without a learned value function. After the policy samples a response y_i and receives feedback f_i, let $c(x_i, f_i)$ denote a re-prompt containing the original prompt and the feedback, together with an instruction to produce a better response that addresses the feedback. SDPO teacher-forces the same response under the original and feedback-conditioned contexts, comparing the resulting next-token distributions over the vocabulary $\mathcal{V}$ (with $v \in \mathcal{V}$ a vocabulary item):

$$p_{i,t}(v) = \pi_\theta(v \mid x_i, y_{i,<t}),$$
$$q_{i,t}(v) = \mathrm{stopgrad}[\pi_\theta(v \mid c(x_i, f_i), y_{i,<t})]\,.$$

The SDPO objective combines GRPO with a Jensen–Shannon divergence (JSD) loss, weighted with a hyperparameter β:

$$\mathcal{L}_{\mathrm{SDPO}} = \mathcal{L}_{\mathrm{GRPO}} + \beta\, \mathbb{E}_{i,t}[D_{\mathrm{JSD}}(p_{i,t}, q_{i,t})]\,, \tag{4}$$

3.2 Token-Level Advantage Estimators

We now formalize the Localized GRPO (L-GRPO) estimator. We assume a task-specific reward constructor has converted the textual judge output into the token rewards $r_{i,t}$ defined above, which we cover in §3.3. Here, we focus on the algorithmic question: how should critic-free GRPO convert these token rewards into token-level advantages?

The reward vector $\{r_{i,t}\}$ contains two signals: its sum S_i, which ranks rollouts within the GRPO group, and the locations and magnitudes of its token rewards, which indicate where the judge placed credit. A token-level advantage estimator must decide how to combine these two signals. We first consider a token-only estimator that uses the reward locations directly, then introduce L-GRPO, which preserves the rollout-level GRPO signal while redistributing credit within each rollout.

Token-only estimator. The simplest estimator ignores rollout returns and normalizes per-token rewards against the *pooled* token statistics of the group,

$$\hat{A}^{\mathrm{tok}}_{i,t} = \frac{r_{i,t} - \mu_r}{\sigma_r + \epsilon}, \tag{5}$$

with μ_r, σ_r the mean and std over all $\{r_{j,s}\}$ in the group. This uses reward locations, but discards the rollout-level comparison that GRPO normally provides. This becomes problematic when rewards are sparse: judges usually score only a small part of the output, such as an error span in a translation or a hallucinated claim in a long answer, while all unmarked tokens receive $r{=}0$.

Consider a simple sparse-reward model: suppose a fraction $p \ll 1$ of tokens are scored with error reward $-c$ and the rest are unscored with reward 0. Then $\mu_r = -pc$ and $\sigma_r = c\sqrt{p(1-p)}$, so for $p \ll 1$ the two token types receive

$$\hat{A}^{\mathrm{tok}}_{i,t}\Big|_{r_{i,t}=0} = \frac{-\mu_r}{\sigma_r + \epsilon} \approx \sqrt{p}, \qquad \hat{A}^{\mathrm{tok}}_{i,t}\Big|_{r_{i,t}=-c} = \frac{-c - \mu_r}{\sigma_r + \epsilon} \approx -\frac{1}{\sqrt{p}}. \tag{6}$$

As rewards become sparse, the unscored-token advantage vanishes, while the magnitude of the error-token advantage grows as $p^{-1/2}$. Thus most tokens, including unscored chain-of-thought tokens, receive a small rollout-independent coefficient rather than a quality-conditioned learning signal, while the few scored tokens carry high-variance updates. We confirm empirically that the token-only estimator is unstable for both MT and factuality (§5).

L-GRPO: broadcast plus redistribution. We instead preserve GRPO's rollout-level broadcast and use the token rewards only to redistribute advantages *within* each rollout. For rollout i, let $\bar{r}_i = S_i/T_i$ be its average token reward and let $d_{i,t} = r_{i,t} - \bar{r}_i$ be token t's deviation from that average. We add the normalized deviation as a localized advantage term:

$$\hat{A}^{\mathrm{LGRPO}}_{i,t} = \underbrace{\frac{S_i - \mu_S}{\sigma_S + \epsilon}}_{\hat{A}^{\mathrm{GRPO}}_i \text{ (broadcast)}} + \alpha\, \underbrace{\frac{d_{i,t}}{\sigma_d + \epsilon}}_{\hat{A}^{\mathrm{loc}}_{i,t}}. \tag{7}$$

Here σ_d is the standard deviation of $\{d_{i,t}\}$ pooled over the judge-graded tokens in the group, and α controls the weighting of the two terms.[2] The localized advantage term is a pure redistribution

[2]We compute $d_{i,t}$ only over the tokens the judge grades and set $d_{i,t}{=}0$ elsewhere, so ungraded tokens, including the chain-of-thought, receive exactly $\hat{A}^{\mathrm{GRPO}}_i$. App. C.2 defines the graded-token set for each task.

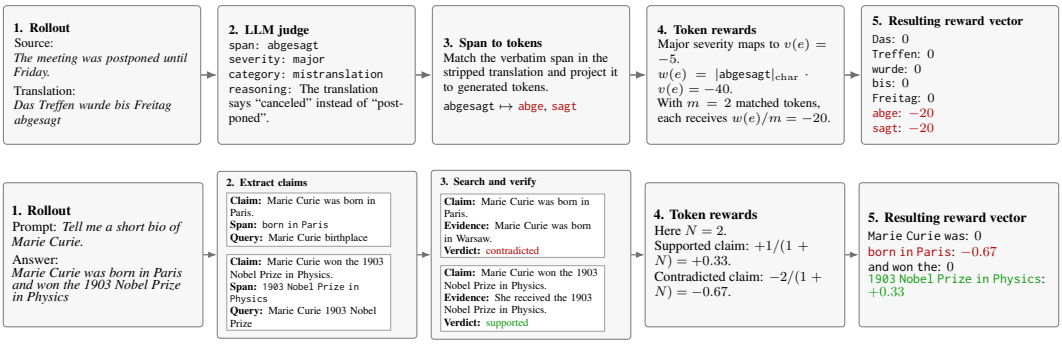


Figure 1: Per-token reward construction for MT (top) and long-form factuality (bottom).

because $\sum_t d_{i,t} = 0$ for every rollout. Therefore it leaves each rollout's mean advantage unchanged:

$$\frac{1}{T_i}\sum_{t=1}^{T_i} \hat{A}_{i,t}^{\text{LGRPO}} = \hat{A}_i^{\text{GRPO}} \qquad \text{for all } \alpha, \tag{8}$$

Thus α changes only how credit is distributed across tokens inside a rollout, moving it toward or away from the spans marked by the judge, while the rollout-level broadcast signal remains unchanged. Importantly, every token, including unscored chain-of-thought tokens the judge never sees, still receives $\hat{A}_i^{\text{GRPO}}$, so chain-of-thought is optimized by the rollout outcome rather than zeroed out, avoiding the atrophy of the token-only estimator.

3.3 Token-level Reward Construction

From spans to tokens. The estimators above assume that each rollout y_i is accompanied by a per-token reward vector. In both applications, however, LLM judges generate span-level rewards: a list of spans s_j (verbatim substrings of y_i) with associated signed rewards w_j. We match each span against the tokenized response using the heuristic described in App. C.1. This heuristic successfully maps $\sim$98% of judged spans uniquely to their corresponding tokens. We distribute w_j uniformly over the matched tokens. If a span cannot be matched uniquely, we distribute its reward uniformly over all answer tokens. Algorithm 1 summarizes the complete procedure below.

Algorithm 1: Span rewards to token rewards.

```
Input: response tokens; judged spans s_j with signed rewards w_j
Output: per-token reward vector r

r = zeros(len(response_tokens))
answer_tokens = tokens_after_cot(response_tokens)
cot_offset = len(response_tokens) - len(answer_tokens)  # answer tokens follow the CoT

for span s, reward w in judged_spans:
    T = map_span(s, answer_tokens)      # local answer-token indices, or UNMAPPED
    if T is UNMAPPED:
        T = range(len(answer_tokens))   # fall back to all answer tokens
    for t in T:
        r[cot_offset + t] += w / len(T)

return r
```

Machine Translation. We now describe the MT-specific spans and reward construction (see Figure 1, top, for an example). We provide the prompt (containing the document to be translated and the target language) and the response (containing the translation with chain-of-thought stripped) to an LLM judge and ask it to return a JSON list of translation error spans according to the MQM taxonomy (Freitag et al., 2021). The full prompt and taxonomy are provided in App. A.2.

The judge annotates each error e with a severity, either major or minor. Following MQM's 5:1 major/minor weighting, we map these labels to signed error weights $v(e) = -5$ for major errors and

Table 1: Mapping from factuality verifier verdicts to signed reward weights.

Verdict	Search evidence	$v(c)$
Supported	Corroborates the claim	$+1$
Contradicted	Refutes the claim	-2
Not-supported	Relevant to the claim but neither supports nor refutes it	-0.25
Not-enough-info	Search results are off-topic and irrelevant to the prompt or claim	0

$v(e)=-1$ for minor errors. For each error e, we compute the span reward $w(e) = \text{char_len}(e.\text{span}) \cdot v(e)$ and distribute it over its tokens using Algorithm 1. Scaling the error weight by the character length of the error span is important for preventing reward hacking. With an unscaled severity weight, we observed the model dropping increasingly large chunks of source content untranslated, an omission error, yet every major omission received the same fixed weight. Character scaling makes this failure mode expensive by penalizing the missing content in proportion to its length.

Long form factuality. We adapt the VeriScore pipeline (Song et al., 2024) to construct per-token rewards in three stages (see Figure 1, bottom, for an example).

Claim extraction: An LLM judge takes the prompt and response (with the chain-of-thought stripped) and extracts independent, atomic factual claims that can be verified through web search. For each claim, the LLM judge also provides a minimal verbatim span from the response that expresses it and a keyword-rich search query for gathering evidence for the claim. The exact extraction prompt is provided in App. B.2, Listing 1.

Web search and claim verification: For each extracted claim, we execute its query using web search and retrieve the top-k results. A second LLM judge is asked to verify the factuality of the claim based on the retrieved search results (complete prompt provided in App. B.2, Listing 2). It assigns a verdict and corresponding signed reward weight $v(c)$ using the mapping in Table 1. Supported claims receive $+1$. We penalize contradicted claims most strongly to encourage the model, when uncertain, to omit or qualify a claim rather than assert a hallucinated fact. Not-supported claims receive only -0.25 because relevant but inconclusive search results provide weaker evidence of an error, while not-enough-info cases receive 0 because the search results are not sufficient to provide any evidence about the claim's factuality.

Per-token rewards: Let N be the number of supported, contradicted, and not-supported claims in a response. We assign each claim c the normalized reward $v(c)/(1+N)$ and distribute it over the tokens corresponding to the claim's verbatim span using Algorithm 1. We exclude not-enough-info claims from the normalization because they reflect search retrieval failures and carry zero weight; including them would shrink the rewards of other claims based on search quality. We add 1 to keep the normalization defined when $N = 0$. Following Chen et al. (2025b), we also add a global relevance reward term that compares each rollout against a reference response generated once by the base policy, discouraging off-topic but factually correct padding; details are provided in App. B.4.

4 Experimental Setup

4.1 Data

Machine translation. We train and evaluate document-level translation from English into ten target languages: Chinese, Arabic, Russian, German, Japanese, Korean, Italian, Ukrainian, Czech, and Hindi. We construct the training set by sampling English documents from Reddit comments (HuggingFaceGECLM, 2026, social), RoyalRoad fiction (OmniAICreator, 2026, literary), CC-News articles (stanford-oval, 2026, news), and English YouTube-Commons transcripts (PleIAs, 2026, speech-style). We retain documents between 300 and 1000 words long. To avoid source duplication, each training document is assigned to one target language, cycling through the ten languages to keep target counts balanced. Evaluation uses the multi-parallel WMT24++ test set (Deutsch et al., 2025), with each English source document evaluated in all ten target languages. The training set contains 17,176 prompts and WMT24++ contributes 1,700 evaluation prompts; source-length statistics are given in Appendix A.1.

Long-form factuality. Following Chen et al. (2025b), we evaluate on four complementary long-form factuality benchmarks. LongFact-Objects (Wei et al., 2024) contains 250 concept- and object-focused prompts designed to elicit multi-paragraph answers. Biography (Min et al., 2023) contains 183 requests for biographies of Wikipedia entities. FactBench-Hard (Bayat et al., 2024) contains 532 challenging information-seeking prompts selected from LMSYS-Chat-1M, while Factory-Hard (Chen et al., 2025a) contains 421 human-verified prompts designed to elicit factual errors. Together, these benchmarks provide 1,386 evaluation prompts.

We construct the training mixture to provide challenging prompts from the same broad evaluation domains. We begin with the 3,706-prompt RL training set of Chen et al. (2025b), which contains synthetically generated prompts similar in domain to LongFact. We find that the base policy already performs strongly on these prompts, so we augment the set with 500 Biography training prompts, 1,500 prompts from Factory-Medium, and 332 prompts from FactBench-Medium. These additions provide harder training examples and improve coverage of the evaluation domains. The resulting training set contains 6,038 prompts. For both tasks, we verify that there is no 10-gram overlap between the training and evaluation sets.

4.2 Model, judges and evaluation metrics

For both tasks, we train Qwen3-8B with GRPO using verl (veRL Team, 2024) and use Gemma 4-31B (Team et al., 2026) as the judge for training. As the judge for evaluation, we use GPT-5.5 and gpt-oss-120b (Agarwal et al., 2025) for MT and factuality respectively to avoid potential bias from using LLMs from same family during training and evaluation. We selected these models as they performed best in our human-agreement studies (see Appendix A.3 and B.3). For factuality, we favor open-weight models due to the judge-intensive nature of factuality evaluation: each response yields roughly 20 claims requiring evidence retrieval and verification. Training and evaluation hyperparameters for GRPO/L-GRPO, PPO, and SDPO are provided in Appendices C.3, C.4, and C.5.

Machine translation. We report two complementary MT quality metrics. First, using GPT-5.5 error spans, we compute a source-normalized density from the same character-weighted span penalties used for training (§1):

$$\rho_d = \frac{\sum_{e \in E_d} |v(e)|\,|e|_{\text{char}}}{|S_{\text{src},d}|_{\text{char}}}.$$

Here, E_d is the set of judged errors in document d, and the denominator is the source-document length in characters. Lower ρ_d is better. For a complementary view from a learned MT metric, we also evaluate using MetricX-24-Hybrid-XXL (Juraska et al., 2024) in reference-based mode (hereafter MetricX-24), which returns an error score in $[0, 25]$, where lower is better.[3] For both metrics, scores are averaged over documents within each language and macro-averaged across languages.

Long-form factuality. For each response, let T be the total number of extracted claims and let S, C, U, and I denote the numbers of supported, contradicted, not-supported, and not-enough-info claims, respectively. Following Chen et al. (2025b), our primary metric is factual precision $P = S/T$, the fraction of all extracted claims supported by retrieved evidence. We also report the reward-aligned score $\mathrm{R}_{\text{fact}} = (S - 2C - 0.25U)/(1 + T)$ as a secondary metric. It mirrors the signed claim weights used during training and excludes not-enough-info claims, which receive zero reward. Both metrics are computed per response and then averaged within each benchmark; overall scores macro-average the four benchmarks. We additionally report mean $S/C/U$ counts.

5 Results

Across both applications we compare L-GRPO against the base Qwen3-8B checkpoint, standard GRPO, two baselines that also use per-token advantages, PPO and SDPO (§3.1), and the naive token-only estimator (§3.2). All RL arms use the same policy initialization, data, reward construction, training grader, and compute budget, differing only in the reward-to-advantage estimator. L-GRPO consistently improves over all of them on both tasks.

[3] Because we evaluate document-level generations, roughly 20% of WMT24++ examples exceed the metric's 1536-token input cap and are excluded from MetricX-24 aggregates; see Appendix A.5 for details.

Table 2: **MT results.** Lower is better for both metrics. * denotes a statistically significant improvement over GRPO ($p < 0.05$ for aggregate comparisons; $p < 0.005$ for language-level comparisons).

Model/Metric	Ar	Cs	De	Hi	It	Ja	Ko	Ru	Uk	Zh	Overall
ρ_d											
Qwen3-8B	0.424	0.610	0.365	0.798	0.311	0.192	0.218	0.346	0.619	0.059	0.394
RL w/ token-only	0.383	0.547	0.312	0.645	0.270	0.161	0.204	0.299	0.545	0.055	0.342
RL w/ PPO	0.426	0.616	0.347	0.596	0.300	0.189	0.212	0.309	0.581	0.057	0.363
RL w/ SDPO	0.399	0.591	0.314	0.686	0.275	0.204	0.206	0.327	0.566	0.065	0.363
RL w/ GRPO	0.356	0.566	0.279	0.583	0.232	0.157	0.192	0.292	0.521	0.049	0.323
RL w/ L-GRPO	**0.326**	**0.509**	**0.266**	**0.530**	**0.215**	**0.156**	**0.172**	**0.280**	**0.496**	**0.045**	**0.299***
METRICX-24											
Qwen3-8B	6.41	10.58	4.08	7.28	5.48	6.35	6.03	6.34	8.58	3.50	6.46
RL w/ token-only	6.34	9.84	3.88	6.91	5.05	6.41	6.15	6.31	8.27	3.56	6.27
RL w/ PPO	6.83	11.02	4.22	7.19	5.53	6.31	6.22	6.46	8.48	3.68	6.59
RL w/ SDPO	6.41	10.37	4.03	7.11	5.29	6.21	5.99	6.21	8.23	3.52	6.34
RL w/ GRPO	6.04	9.94	3.65	6.89	5.12	6.31	6.30	5.82	8.14	3.56	6.18
RL w/ L-GRPO	**5.77**	**9.49**	**3.58**	**6.55**	**4.73***	**6.15**	**5.80***	**5.80**	**7.43***	**3.48**	**5.88***

Machine translation. On document-level MT, L-GRPO reduces the macro-averaged error density ρ_d under GPT-5.5 judging by 7.2%, from 0.323 to 0.299, and the METRICX-24 error score by 4.8%, from 6.18 to 5.88 (Table 2). Both aggregate improvements are statistically significant. At the language level, L-GRPO yields lower observed error scores in all ten languages under both metrics. Together, the significant aggregate gains and consistent per-language directions show that the judge's span locations provide optimization signal beyond its scalar rollout score.

Long-form factuality. Compared to GRPO, L-GRPO increases macro-averaged factual precision across the four benchmarks from 66.2% to 72.5% (9.6% relative; Table 3). Precision improves on all four benchmarks, with significant gains on FactBench-Hard and Biography. The reward-aligned R_{fact} improves even more, increasing from 44.8% to 51.5% (15% relative). These gains reflect more supported claims per response (13.0 to 13.7) and fewer contradicted claims (1.14 to 1.00).

In both tasks, these gains are not an artifact of checkpoint selection: L-GRPO outperforms GRPO at all evaluated checkpoints after an initial warmup, on MT under both metrics and on factuality for both precision and R_{fact} (Appendices A.4 and B.5).

Alternative fine-grained baselines. Despite constructing fine-grained per-token advantages, both PPO and SDPO underperform the standard GRPO baseline on both tasks. On MT, SDPO and PPO reach ρ_d/METRICX $-$ 24 of 0.363/6.34 and 0.363/6.59, both worse than GRPO (0.323/6.18). On factuality, SDPO and PPO reach 55.9% and 60.5% factual precision, both below GRPO (66.2%).

Critic diagnostics suggest one explanation for PPO's underperformance (Appendix C.6): its value function only imperfectly predicts reward-to-go, and this value-estimation error degrades its token-level advantages. GRPO and L-GRPO avoid it, obtaining a prompt-specific baseline directly from the rollout group rather than through a learned value function. For SDPO, the self-distillation signal is an indirect proxy for response quality: the feedback-conditioned teacher's probability shifts need not localize to the judge-identified error spans. Recent work (Zhang et al., 2026; Li et al., 2026; Wang et al., 2026b) also report that this signal is a biased surrogate for the reward objective that can oppose reward-improving updates once the policy is capable .

Token-only estimator. RL with the token-only estimator is unstable on both tasks. After initial gains, training reward collapses, KL rises, entropy falls, and evaluation metrics deteriorate (Appendix Figures 3, 5, and 8). Even at its best pre-collapse checkpoint, it trails the final GRPO and L-GRPO checkpoints on both tasks under all metrics (Tables 2 and 3). This confirms L-GRPO's central design choice: localized rewards are useful for redistributing credit within a rollout, but GRPO's rollout-level advantage remains necessary for stable training.

Sample efficiency. Our main factuality experiments use a rollout group size of $G = 12$. To compare L-GRPO and GRPO on sample efficiency, we additionally rerun both with $G \in \{4, 6\}$.

Table 3: **Long-form factuality results.** Factual precision P and $\mathrm{R}_{\mathrm{fact}}$ (%) per benchmark; Overall macro-averages these and the $S/C/U$ counts. Higher is better for P, $\mathrm{R}_{\mathrm{fact}}$, S; lower for C, U. * marks significant improvement over GRPO ($p < 0.0125$). Per-benchmark $S/C/U/I$ are available in Appendix Table 9.

Model	LongFact		FactBench-Hard		Factory-Hard		Biography		Overall		
	P	$\mathrm{R}_{\mathrm{fact}}$	P	$\mathrm{R}_{\mathrm{fact}}$	P	$\mathrm{R}_{\mathrm{fact}}$	P	$\mathrm{R}_{\mathrm{fact}}$	$S/C/U$	P	$\mathrm{R}_{\mathrm{fact}}$
Qwen3-8B	70.0	42.3	63.4	37.3	33.7	4.6	21.1	-45.2	12.7 / 3.6 / 5.3	47.1	9.8
RL w/ token-only	73.4	55.2	69.5	47.3	37.9	15.0	32.2	-16.9	14.8 / 2.5 / 4.7	53.3	25.1
RL w/ PPO	73.8	52.2	69.2	44.9	48.0	26.3	50.8	5.8	**22.4** / 3.5 / 6.3	60.5	32.3
RL w/ SDPO	81.1	61.9	70.6	46.6	37.0	10.0	35.0	-10.8	11.5 / 1.8 / 3.1	55.9	26.9
RL w/ GRPO	87.8	73.7	78.1	58.7	56.7	30.5	42.3	16.2	13.0 / 1.1 / **1.6**	66.2	44.8
RL w/ L-GRPO	**89.2**	**77.8***	**80.4***	**62.3***	**59.2**	**38.6***	**61.3***	**27.3**	13.7*/**1.0***/1.7	**72.5***	**51.5***

We evaluate intermediate checkpoints from all six runs every ten training steps and plot factual precision against the number of rollouts sampled up to each checkpoint. At each group size, L-GRPO reaches GRPO's final precision with fewer rollouts: 35.7%, 43.3%, and 23.8% fewer for $G = 4, 6, 12$, respectively (Figure 2). Thus, L-GRPO improves not only final quality but also the rate at which the policy learns from sampled generations.

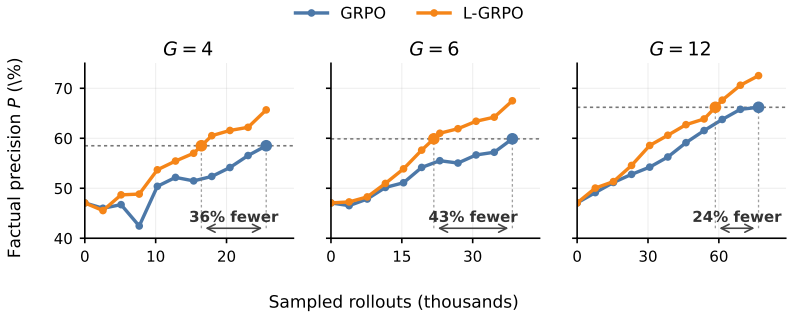


Figure 2: **Sample efficiency on long-form factuality.** Macro-averaged factual precision vs. cumulative sampled rollouts, one panel per rollout group size G. The dotted line marks GRPO's final precision; arrows show the rollouts L-GRPO saves to reach it.

6 Conclusion

We introduced L-GRPO, a critic-free advantage estimator that enables GRPO to use localized rewards from LLM judges. Standard GRPO reduces token-level rewards to a scalar rollout return and broadcasts the resulting advantage to every token. L-GRPO preserves this group-relative signal while adding a mean-zero redistribution term derived from the locations of the token rewards. It therefore changes where advantages are assigned within a response without changing the rollout's mean advantage or introducing a learned value function.

Under controlled comparisons with Qwen3-8B, L-GRPO improves over GRPO on two distinct generation tasks. On document-level MT, it reduces error density by 7.2%; on long-form factuality, it improves factual precision by 9.6% relative, while producing more supported claims. It also outperforms the PPO and SDPO baselines on both tasks, and reaches GRPO's final factual precision with 34.3% fewer sampled rollouts on average. These findings suggest that LLM judges should be treated not only as sources of scalar scores, but also as sources of structured optimization signal. Whenever an automatic grader can identify which parts of a response are correct or erroneous, collapsing its output to a single reward discards useful supervision. L-GRPO provides a general way to retain that information within critic-free policy optimization, with potential applications to grounding, summarization faithfulness, and instruction following.

AI Use Statement

In this work, we used generative AI tools: Anthropic's Claude (including Claude Code) and OpenAI's GPT models, for several tasks that require disclosure. Among required-disclosure tasks, we used these tools to help implement methods and experiment code, to obtain feedback on the design of our methodology and experiments, and to help interpret experimental results. We did not use generative AI tools to generate synthetic datasets, to clean or reformat datasets, to develop the paper's conceptual framework, to formulate mathematical claims, or to propose or refine our research hypotheses; these were carried out by the authors. The remaining required-disclosure tasks: assisting in the writing of proofs, translation, qualitative and thematic data analysis, and formulating survey or interview questions, are not applicable to this work.

Additionally, among recommended-disclosure tasks, we used generative AI tools to draft and edit portions of the manuscript for clarity and concision, to search for, identify, and summarize related work, and to help create figures (e.g., the reward-construction pipeline diagram) and format bibliographic references.

We have reviewed all AI-assisted work. AI-assisted code was verified and tested by the authors; AI-assisted text and results interpretations were checked for accuracy by the authors; and all references identified with AI assistance were verified against their original sources. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

Reproducibility Statement

The details needed to reproduce our results are distributed across the main text and appendix. The L-GRPO estimator and its relation to GRPO are specified in §3.2 (Eq. 7); the conversion of judge outputs into per-token rewards is given in §3.3 and Algorithm 1, with the span-to-token matching heuristic in Appendix C.1. Complete judge prompts and reward mappings appear in Appendix A.2 (MT/MQM) and Appendix B.2 (factuality extraction and verification), and the relevance term in Appendix B.4. Datasets, splits, and filtering are described in §4 and Appendix A.1, and all training, evaluation, and baseline hyperparameters for GRPO/L-GRPO, PPO, and SDPO are provided in Appendices C.3, C.4, and C.5, together with the metric definitions and significance-testing procedures in Appendices A.4 and B.5.

Some components rely on internal infrastructure and are not released directly: the web-search backend used for factuality claim verification, the stack used to serve the judge models during training and evaluation, and the training data itself, which is drawn from publicly available corpora under their own licenses and is therefore not redistributed. None of these is essential to reproducing the method. The judge models are publicly available (Gemma 4-31B and gpt-oss-120b are open-weight; GPT-5.5 is a commercial API); our code accesses the search backend and judge servers through narrow interfaces—a query-to-results search function and an OpenAI-compatible chat endpoint—that can be backed by any equivalent web-search API and standard open-weight serving framework.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Farima Fatahi Bayat, Lechen Zhang, Sheza Munir, and Lu Wang. Factbench: A dynamic benchmark for in-the-wild language model factuality evaluation, 2024. URL https://arxiv.org/abs/2410.22257.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Enhancing reinforcement learning with dense rewards from language model critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9119–9138. Association for Computational Linguistics, 2024.

Alex J. Chan, Hao Sun, Samuel Holt, and Mihaela van der Schaar. Dense reward for free in reinforcement learning from human feedback. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.00782.

Mingda Chen, Yang Li, Xilun Chen, Adina Williams, Gargi Ghosh, and Scott Yih. Factory: A challenging human-verified prompt set for long-form factuality, 2025a. URL https://arxiv.org/abs/2508.00109.

Xilun Chen, Ilia Kulikov, Vincent-Pierre Berges, Barlas Oğuz, Rulin Shao, Gargi Ghosh, Jason Weston, and Wen-tau Yih. Learning to reason for factuality. *arXiv preprint arXiv:2508.05618*, 2025b.

Daniel Deutsch, Eleftheria Briakou, Isaac Rayburn Caswell, Mara Finkelstein, Rebecca Galor, Juraj Juraska, Geza Kovacs, Alison Lui, Ricardo Rei, Jason Riesa, Shruti Rijhwani, Parker Riley, Elizabeth Salesky, Firas Trabelsi, Stephanie Winkler, Biao Zhang, and Markus Freitag. WMT24++: Expanding the language coverage of WMT24 to 55 languages & dialects. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12257–12284, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.634. URL https://aclanthology.org/2025.findings-acl.634/.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In *Proceedings of the Eighth Conference on Machine Translation*, pp. 1066–1083, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.wmt-1.100. URL https://aclanthology.org/2023.wmt-1.100/.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics (TACL)*, 2021.

Nuno M. Guerreiro, Ricardo Rei, Daan van Stigt, Luisa Coheur, Pierre Colombo, and André F. T. Martins. xCOMET: Transparent machine translation evaluation through fine-grained error detection. *Transactions of the Association for Computational Linguistics*, 12:979–995, 2024. doi: 10.1162/tacl_a_00683. URL https://aclanthology.org/2024.tacl-1.54/.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in reinforcement learning for large language models. *arXiv preprint arXiv:2505.23564*, 2025.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.20802.

HuggingFaceGECLM. REDDIT_comments. https://huggingface.co/datasets/HuggingFaceGECLM/REDDIT_comments, 2026. Hugging Face dataset repository; accessed 2026-07-16.

Juraj Juraska, Daniel Deutsch, Mara Finkelstein, and Markus Freitag. MetricX-24: The Google submission to the WMT 2024 metrics shared task. In *Proceedings of the Ninth Conference on Machine Translation (WMT)*, 2024.

Ahrii Kim. RUBRIC-MQM: Span-level LLM-as-judge in machine translation for high-end models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 147–165, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-288-6. doi: 10.18653/v1/2025.acl-industry.12. URL https://aclanthology.org/2025.acl-industry.12/.

Tom Kocmi and Christian Federmann. GEMBA-MQM: Detecting translation quality error spans with GPT-4. In *Proceedings of the Eighth Conference on Machine Translation (WMT)*, 2023.

Alon Lavie, Greg Hanneman, Sweta Agrawal, Diptesh Kanojia, Chi-Kiu Lo, Vilém Zouhar, Frederic Blain, Chrysoula Zerva, Eleftherios Avramidis, Sourabh Deoghare, Archchana Sindhujan, Jiayi Wang, David Ifeoluwa Adelani, Brian Thompson, Tom Kocmi, Markus Freitag, and Daniel Deutsch. Findings of the WMT25 shared task on automated translation evaluation systems: Linguistic diversity is challenging and references still help. In *Proceedings of the Tenth Conference on Machine Translation*, pp. 436–483, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-341-8. doi: 10.18653/v1/2025.wmt-1.24. URL https://aclanthology.org/2025.wmt-1.24/.

Gengsheng Li, Tianyu Yang, Junfeng Fang, Mingyang Song, Mao Zheng, Haiyun Guo, Dan Zhang, Jinqiao Wang, and Tat-Seng Chua. Unifying group-relative and self-distillation policy optimization via sample routing. *arXiv preprint arXiv:2604.02288*, 2026. URL https://arxiv.org/abs/2604.02288.

Sheng-Chieh Lin, Luyu Gao, Barlas Oguz, Wenhan Xiong, Jimmy Lin, Wen-tau Yih, and Xilun Chen. FLAME: Factuality-aware alignment for large language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=zWuHSIALBh.

Qingyu Lu, Baopu Qiu, Liang Ding, Kanjian Zhang, Tom Kocmi, and Dacheng Tao. Error analysis prompting enables human-like translation evaluation in large language models. In *Proceedings of the Eighth Conference on Machine Translation (WMT)*, 2023.

Qingyu Lu, Liang Ding, Kanjian Zhang, Jinxia Zhang, and Dacheng Tao. MQM-APE: Toward high-quality error annotation predictors with automatic post-editing in LLM translation evaluators. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2401.00396.

OmniAICreator. RoyalRoad-1.61M. https://huggingface.co/datasets/OmniAICreator/RoyalRoad-1.61M, 2026. Hugging Face dataset repository; accessed 2026-07-16.

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

PleIAs. YouTube-Commons. https://huggingface.co/datasets/PleIAs/YouTube-Commons, 2026. Hugging Face dataset repository; accessed 2026-07-16.

Miguel Moura Ramos, Tomás Almeida, Daniel Vareta, Filipe Azevedo, Sweta Agrawal, Patrick Fernandes, and André F. T. Martins. Fine-grained reward optimization for machine translation using error severity mappings. *Transactions of the Association for Computational Linguistics (TACL)*, 2025.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yixiao Song, Yekyung Kim, and Mohit Iyyer. VeriScore: Evaluating the factuality of verifiable claims in long-form text generation. In *Findings of the Association for Computational Linguistics (EMNLP Findings)*, 2024. arXiv:2406.19276.

stanford-oval. CC-News. https://huggingface.co/datasets/stanford-oval/ccnews, 2026. Hugging Face dataset repository; accessed 2026-07-16.

Hongze Tan, Zihan Wang, Jianfei Pan, Jinghao Lin, Hao Wang, Yifan Wu, Tao Chen, Zhihang Zheng, Zhihao Tang, and Haihua Yang. GTPO and GRPO-S: Token and sequence-level reward shaping with policy entropy. *arXiv preprint arXiv:2508.04349*, 2025.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

veRL Team. veRL: Volcano Engine Reinforcement Learning for LLMs. https://github.com/volcengine/verl, 2024. Software repository; accessed 2026-07-16.

Hanyang Wang, Lu Wang, Chaoyun Zhang, Tianjun Mao, Si Qin, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Text2Grad: Reinforcement learning from natural language feedback. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2505.22338.

Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser $\neq$ better: Limits of on-policy self-distillation for continual post-training. *arXiv preprint arXiv:2607.01763*, 2026b. URL https://arxiv.org/abs/2607.01763.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. Long-form factuality in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhepei Wei, Xiao Yang, Kai Sun, Jiaqi Wang, Rulin Shao, Sean Chen, Mohammad Kachuee, Teja Gollapudi, Tony Liao, Nicolas Scheffer, Rakesh Wanga, Anuj Kumar, Yu Meng, Wen-tau Yih, and Xin Luna Dong. TruthRL: Incentivizing truthful LLMs via reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.25760.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.01693.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Eunseop Yoon, Hee Suk Yoon, Soohwan Eom, Gunsoo Han, Daniel Wontae Nam, Daejin Jo, Kyoung-Woon On, Mark Hasegawa-Johnson, Sungwoong Kim, and Chang D. Yoo. TLCR: Token-level continuous reward for fine-grained reinforcement learning from human feedback. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024. arXiv:2407.16574.

Yubo Zhang, Xinhong Ma, Zezhong Tan, and Ziqiang Dong. I-sdpo: Instance-level adaptive self-distillation policy optimization. *arXiv preprint arXiv:2608.12957*, 2026. URL https://arxiv.org/abs/2608.12957.

A Machine Translation

A.1 Data Statistics

The training set has a median source length of 508 words (95th percentile 942 words). WMT24++ evaluation prompts have a median source length of 85 words (95th percentile 882 words).

A.2 Judge prompt and taxonomy

We use the following LLM judge prompt for training and evaluation. The fields `{target}`, `{source}`, and `{translation}` are placeholders for the target language, source text to be translated, and the generated translation, respectively. The prompt uses MQM-style error categories and severities.

```
You are an expert evaluator of English->{target} machine translation. Find EVERY error as a span.
There are two kinds:
1. TARGET error: a span of the {target} translation that is wrong (mistranslation, addition, grammar, spelling,
     terminology, register, or style).
2. OMISSION: source content missing from the translation -- return the dropped ENGLISH SOURCE text as the span.

For each error return a JSON object with:
- "side": "target" for a {target}-translation error, or "source" for an omission (the missing English source text
     ).
- "span": the exact, verbatim, MINIMAL substring (copied character-for-character) -- from the {target}
     translation when side="target", from the English source when side="source".
- "severity": "major" if it substantially changes meaning or harms understanding, otherwise "minor".
- "category": one of: mistranslation, addition, omission, grammar, spelling, terminology, register, style.
- "note": a brief English explanation of what is wrong and why.
- For side="source" omissions ONLY, also add "anchor_after": the minimal verbatim {target} span (2-5 words,
     copied character-for-character from the translation) IMMEDIATELY AFTER the position where the dropped
     content should have been inserted (use "" only if it belongs at the very end), and "anchor_before": the
     minimal verbatim {target} span just BEFORE that position (use "" if at the very start).

Return ONLY a JSON object, no other text:
{"errors": [{"side": "target", "span": "...", "severity": "major", "category": "mistranslation", "note": "..."},
     {"side": "source", "span": "...", "severity": "minor", "category": "omission", "note": "...", "anchor_after
     ": "...", "anchor_before": "..."}]}
If the translation has no errors, return {"errors": []}.

Source (English):
{source}

Translation ({target}):
{translation}

JSON:
```

Omissions have no target-side error span because the erroneous content is absent from the translation. The judge therefore returns a verbatim target-side `anchor_after` immediately following where the missing content should have appeared, and we treat that anchor as the error span when constructing per-token rewards. Its signed contribution is still scaled by the character length of the omitted source span. If the omission belongs at the end of the translation, we use `anchor_before` instead. We cap the magnitude of any span reward at $|w(e)| \leq 8000$ so that a single large omission cannot dominate the update.

A.3 Translation Judge Agreement with Human Annotations

The WMT24 Error Span Annotation (ESA) dataset contains system translations annotated by humans with error spans and major/minor severities. We sample 1,000 translated segments for each of the six language pairs that overlap our target languages: English to Czech, Japanese, Chinese, Hindi, Ukrainian, and Russian. We evaluate five candidate models as LLM translation judges against these human annotations.[4] All judges receive the prompt listed in Appendix A.2. Models are used with thinking enabled; gpt-oss-120b uses medium reasoning effort.

[4] WMT24 ESA annotations are from https://github.com/wmt-conference/wmt24-news-systems, file `txt/wmt24_esa_original.jsonl`.

Table 4: **Judge agreement with human-labeled translation errors.** Span F1 measures span-level agreement, and segment correlation is Spearman correlation between judge-derived and human segment-level error scores.

Model	Type	Span F1 ↑	Seg. corr. ↑
Gemma 4 31B	open	**0.239**	**0.295**
GPT-5.5	API	0.221	0.291
Gemini 3.1 Pro	API	0.212	0.288
Gemini 3.5 Flash	API	0.190	0.287
gpt-oss-120b	open	0.167	0.244

Table 5: **Reference-context ablation.** Span F1 on translations with available WMT24++ references. Δ is reference-based minus reference-free; adding a reference yields no consistent improvement.

Model	Reference-free	Reference-based	Δ
Gemma 4 31B	0.199	0.194	-0.005
Gemini 3.1 Pro	0.176	0.179	+0.004
Gemini 3.5 Flash	0.179	0.178	-0.001

We measure agreement with human annotations at the span level and the translation level. At the span level, we report Span F1 using the WMT25 Task-2 character-level scorer.[5] The scorer expands the human and judge error spans into per-character severity labels, gives full credit to overlapping characters with matching severities and partial credit to severity mismatches, and computes precision, recall, and F1 from the matched character counts. At the segment level, we compute Spearman correlation between each judge's overall error score and the human error score, obtained by negating the ESA quality score. Results are reported in Table 4.

We additionally test whether including a human reference in the judge prompt improves span agreement. On the subset with available WMT24++ references, we evaluate identical translations with and without reference context. Table 5 shows no consistent improvement, so all training and evaluation judging is reference-free.

A.4 Evaluation Trends and Statistical Significance

Table 2 reports the final step-150 GRPO and L-GRPO checkpoints, the best observed token-only checkpoint, and the best SDPO checkpoint before its late deterioration. To assess whether the main gains depend on checkpoint selection, we evaluate all estimators every ten training steps using both test-set metrics. As shown in Figure 3, by step 40, L-GRPO outperforms GRPO under both GPT-5.5 ρ_d and MetricX-24 and remains better at every subsequent checkpoint. Although the trajectories are not strictly monotonic, their sustained ordering shows that the final-checkpoint gains reflect a consistent training trend rather than a favorable checkpoint selected in isolation.

The token-only estimator improves initially, reaching its lowest ρ_d at step 60 (0.342) and its lowest MetricX-24 score at step 50 (6.24). Both metrics reverse after step 70 and deteriorate sharply through the final available checkpoint at step 110 (0.753 and 7.35, respectively). For a checkpoint-consistent comparison, Table 2 reports step 60 under both metrics. The token-only scores use the paired non-truncated subset available for that checkpoint and are not included in the final-checkpoint significance tests below.

SDPO improves more gradually, reaching its best scores at step 100 ($\rho_d = 0.363$ and MetricX-24 $= 6.34$) before both metrics worsen at step 110. Table 2 therefore reports step 100 for this baseline. Even at that checkpoint, it trails both GRPO and L-GRPO under both metrics.

Final-checkpoint significance. Because WMT24++ is multi-parallel, translations into different languages share the same English sources. We therefore estimate uncertainty with 10,000 paired

[5] We use `get_char_f1` from the WMT25 Metrics shared-task scorer: https://github.com/wmt-conference/wmt25-mteval, file `scripts/scoring/task2/scoring_esa.py`.

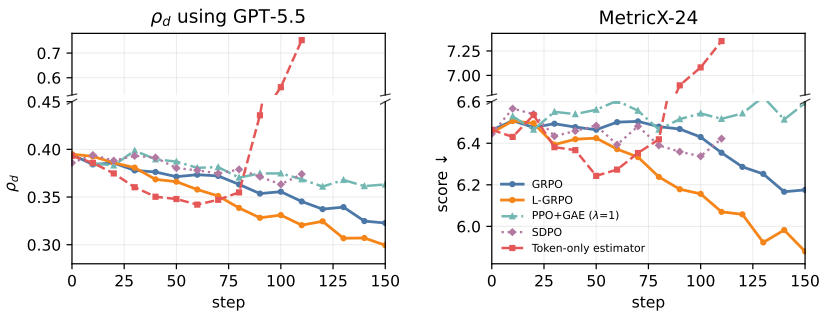


Figure 3: **MT evaluation over training.** Test-set ρ_d using GPT-5.5 as judge (left) and MetricX-24 (right); lower is better for both metrics. From step 40 onward, L-GRPO outperforms GRPO at every checkpoint under both metrics. The token-only estimator improves initially but degrades sharply after step 70 and terminates at step 110. SDPO reaches its best scores at step 100 and then deteriorates under both metrics.

Table 6: **MT significance by target language.** Relative error reductions from GRPO to L-GRPO at step 150; positive values favor L-GRPO. p-values use paired source-level randomization. * denotes significance after Bonferroni correction over the ten languages within that metric ($p < 0.005$). The Overall row reports the relative reduction of the macro-averaged error scores (as in Table 2), not the mean of the per-language reductions; it is tested once and excluded from the language-level correction.

	GPT-5.5 ρ_d		MetricX-24	
Language	Error reduction (%)	p	Error reduction (%)	p
Arabic	8.6	.1134	4.5	.1665
Czech	10.0	.0081	4.5	.0876
German	4.7	.4846	1.9	.5470
Hindi	9.1	.1101	4.9	.0053
Italian	7.4	.2209	7.6	.0002*
Japanese	0.7	.9309	2.4	.1356
Korean	10.7	.0892	7.9	.0007*
Russian	4.2	.7015	0.2	.9321
Ukrainian	4.8	.3842	8.7	.0004*
Chinese	7.1	.4856	2.2	.3156
Overall	7.2	.0002	4.8	$<$ .0001

bootstrap resamples over English-source clusters, retaining all available language versions of a sampled source. The resulting 95% confidence intervals for the macro-averaged error reduction are 3.5–10.9% under GPT-5.5 ρ_d and 3.1–6.5% under MetricX-24. We compute two-sided p-values using 100,000 paired source-level randomizations. Both macro-level gains are significant ($p < 0.001$).

Table 6 reports the corresponding language-level tests. We apply Bonferroni correction over the ten languages within each metric, giving a significance threshold of $p < 0.005$. The MetricX-24 improvements for Italian, Korean, and Ukrainian are significant; none of the GPT-5.5 language-level comparisons reach corrected significance. The remaining point estimates all favor L-GRPO but do not reach significance at this test-set size.

A.5 MetricX-24 Coverage

MetricX-24-Hybrid-XXL has a 1536-token context budget for the source, translation, and reference together. In our document-level setting, the complete MetricX-24 input exceeds this limit for roughly 20% of WMT24++ examples. We exclude these examples because scores computed from truncated inputs are unreliable.

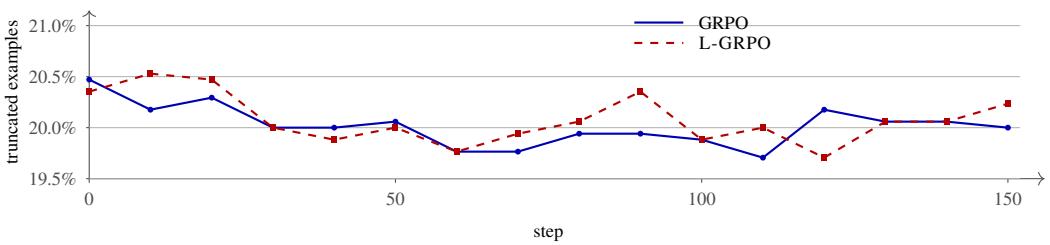


Figure 4: **METRICX-24 coverage over training.** Fraction of test examples excluded because the complete METRICX-24 input exceeds its 1536-token context window. The rate stays close to 20% and exhibits no systematic trend for either GRPO or L-GRPO.

The improving METRICX-24 trajectory in Figure 3 could otherwise be confounded by a systematic change in this exclusion rate over training. Figure 4 shows that the rate instead remains close to 20% for both GRPO and L-GRPO, with no systematic trend. The improvement in test-set METRICX-24 therefore cannot be explained by changing coverage. For the final comparison in Table 2, GRPO and L-GRPO are evaluated on the common subset of examples for which the complete METRICX-24 inputs from both checkpoints fit within the context window, so the two systems are scored on exactly the same source examples. Each additional baseline (token-only, PPO+GAE, and SDPO) is likewise scored on its own non-truncated subset paired with the L-GRPO checkpoint; because truncation patterns differ across systems, these per-baseline subsets differ slightly, so each system-versus-L-GRPO METRICX-24 comparison is paired but the table rows do not all share a single document set.

B LONG-FORM FACTUALITY

B.1 BENCHMARK SELECTION

Chen et al. (2025b) additionally evaluate FAVA and AlpacaFact. We omit them because we could not locate the selection scripts from Lin et al. (2024) used to construct their evaluation subsets and therefore could not reproduce the exact prompt sets.

B.2 FACTUALITY GRADING PIPELINE AND IMPROVEMENTS OVER VERISCORE

Our factuality grader follows VeriScore's extract–search–verify pipeline, with a few refinements that improve its reliability and make its claim-level judgments usable as token-level rewards for L-GRPO.

B.2.1 CLAIM EXTRACTION

VeriScore segments a response into sentences and invokes a fine-tuned claim extractor independently for each sentence. Each call receives the sentence of interest together with up to three preceding sentences, one following sentence, and the prompt. In pilot training runs, we found that this decomposition introduced extraction and decontextualization errors when a claim depended on information elsewhere in the response. Modern long-context LLMs, including the judges used in our experiments, can process the complete answer directly. We therefore number its sentences and extract all claims in a single call conditioned on both the prompt and the full response. We found this whole-response extraction substantially more reliable while also requiring fewer LLM calls.

We make two further changes to the extractor output. First, VeriScore submits each extracted claim directly to its search engine. This can retrieve poor evidence when the claim is long or verbose, so we ask the extractor to produce a compact, keyword-rich search query for each claim. In our pilot runs, these queries returned more relevant evidence than searching with the claim itself. Second, VeriScore does not identify the exact response span corresponding to each extracted claim. We therefore require a minimal verbatim span for every claim, allowing its reward to be mapped back to the corresponding generated tokens. The complete extraction prompt is provided in Listing 1.

Listing 1: Whole-answer claim extraction (per claim: claim, search query, and verbatim span).

```
You extract atomic, verifiable, decontextualized factual claims AND a web-search query to verify each.

Question: {question}

Below are consecutive sentences from a response to that question, each tagged with an index [S#]. For EACH
      sentence, extract the atomic factual claims it asserts THAT CAN BE VERIFIED THROUGH WEB SEARCH against
      reliable external sources. Use the whole response and the question to decontextualize each claim (resolve
      pronouns/references) so it stands alone and is checkable.

Do NOT extract statements that are not checkable via web search (e.g. a mathematical derivation, the result of a
      calculation, a definition, or a claim about the response's own reasoning), opinions, questions,
      instructions, hypotheticals, or predictions. If a sentence asserts no web-verifiable claim, use an empty
      list for it.

For EACH claim also produce a "query": the best keyword-rich web-search query to find evidence, including the
      specific disambiguating details (named entity, dates, numbers, titles) and anchored to the main subject of
      the response.

For EACH claim also produce a "span": the minimal contiguous, verbatim substring of THAT sentence that contains
      the fact (copied character-for-character); if none captures it, use the whole sentence.

Sentences:
{numbered_sentences}

Respond with ONLY a JSON object mapping each sentence number (as a string) to a list of {"claim":..., "query":...,
       "span":...} objects (empty list if none).
```

B.2.2 SEARCH

We submit each generated query to an internal open-web search engine and request up to ten results per claim during both training and evaluation. For each result, we pass its title, URL, and complete returned snippet to the verifier; we apply no additional token cap to the evidence.

B.2.3 CLAIM VERIFICATION

Each response yields roughly 20 claims on average, so independently verifying every claim across every rollout would make online reward computation prohibitively computationally expensive. During training, we therefore batch up to four claims from the same response into each verification call. We chose this batch size by comparing verdict agreement against single-claim verification: batching four claims caused minimal degradation while cutting judge calls roughly $4\times$, whereas batching eight introduced noticeable cross-claim interference. Evaluation is performed offline, so we verify one claim per call for maximum fidelity. The verifier receives each claim with its retrieved search results and uses the original prompt only to resolve ambiguous references. It then assigns one of the four verdicts defined in §1. The complete verification prompt is provided in Listing 2.

Listing 2: Four-way claim verification against the claim's own retrieved evidence.

```
You judge each claim against its own web search results, choosing ONE decision:
 - "supported": one or more results directly support the claim (all its parts) and none contradict it.
 - "contradicted": one or more results directly contradict the claim (assert something incompatible: a different
      date, place, number, name, or outcome).
 - "not_supported": the results DO cover the claim's subject (on-topic, would be expected to mention this fact)
      but do NOT corroborate the specific claim.
 - "not_enough_info": the results are missing, off-topic, or do not cover the claim's subject at all (a genuine
      retrieval gap).
KEY DIFFERENCE: "not_supported" = relevant coverage exists but does not back the claim; "not_enough_info" = no
      relevant coverage at all.

The claims below were all extracted from ONE response to this user request:
{question}
Use this only to interpret ambiguous references; do NOT treat it as evidence.

Claim 1: {claim_1}
Search results for claim 1:
{search_results_1}

...

Claim K: {claim_K}
Search results for claim K:
{search_results_K}
```

Table 7: **Agreement with Factory human factuality judgments under self-retrieved evidence.** An abstention is counted as an incorrect prediction. Intervals are 95% grouped-bootstrap confidence intervals for macro-F_1.

	Factual			Nonfactual				
Model	P	R	F_1	P	R	F_1	Macro-F_1	95% CI
Gemini 3.1 Pro	.778	.648	.707	.727	.679	.702	.705	[.667, .737]
GPT-5.5	.741	.676	.707	.723	.645	.682	.695	[.659, .729]
Gemma 4 31B	.724	.694	.709	.738	.614	.670	.690	[.652, .720]
gpt-oss-120b	.730	.674	.701	.699	.640	.668	.684	[.649, .718]
Gemini 3.5 Flash	.746	.671	.706	.705	.548	.616	.661	[.625, .693]

```
Respond with ONLY a JSON array, one object per claim in order:
[{"claim": 1, "reasoning": "<1 sentence citing the results>", "decision": "supported"}, ...].
```

B.3 Factuality Verifier Agreement with Human Judgments

We evaluate how closely the factuality pipeline reproduces human judgments. We use the human annotations released with Factory. Each annotated example contains the original prompt, model response, gold atomic claims derived from the response, their factuality labels, and annotator evidence when available. Labels are FACTUAL, NONFACTUAL, or INCONCLUSIVE. We retain the 389 evidence-backed NONFACTUAL claims and randomly sample 389 FACTUAL claims, yielding a balanced set of 778 claims. We exclude INCONCLUSIVE: only 5.2% of these claims have annotator evidence, indicating that this label primarily records retrieval failure rather than a factuality decision.

For each candidate model, we run the factuality pipeline described in Appendix B.2 with one change: during extraction, we provide the gold human-edited claims instead of asking the model to identify claims from the response. The model still generates each claim's search query and localization span; retrieval and four-way verification are otherwise unchanged. We map `supported` to FACTUAL, map `contradicted` and `not_supported` to NONFACTUAL, and treat `not_enough_info` as an abstention that is scored as incorrect. Macro-F_1 is computed over the two gold classes. We obtain 95% confidence intervals from 1,000 bootstrap resamples over response groups, keeping claims from the same response together; pairwise comparisons reuse the same resamples.

Table 7 shows that the four strongest models, including our training and evaluation verifiers, are statistically indistinguishable. Relative to Gemini 3.1 Pro, the paired macro-F_1 differences are 0.010 for GPT-5.5 (95% CI $[-0.021, 0.039]$), 0.015 for Gemma 4 31B ($[-0.012, 0.040]$), and 0.020 for gpt-oss-120b ($[-0.011, 0.049]$). Only the 0.043 difference from Gemini 3.5 Flash is significant ($[0.015, 0.069]$). Factual-claim F_1 varies little across models (0.701–0.709); most variation comes from nonfactual-claim recall (0.548–0.679). Excluding abstentions raises scores but leaves the substantive conclusion unchanged.

This study evaluates retrieval and verification but not unconstrained claim extraction, since claims are pinned to human annotations. It also uses one run per model and one claim per verification call, whereas online training batches up to four claims. Finally, results depend on the search backend and retrieval date.

B.4 Auxiliary Relevance Judge

Claim-level factuality alone leaves an important reward-hacking strategy open: the policy can accumulate reward by producing unrelated or tangential facts that are individually correct but do not answer the prompt. We therefore add an auxiliary relevance term, following Chen et al. (2025b), that compares each rollout against a fixed reference answer for the same prompt.

Before RL training, we use the initial Qwen3-8B policy to generate one reference answer y_{ref} for each of the 6,038 training prompts. These answers are cached and held fixed throughout optimization. For a rollout y_i, the relevance judge returns $b_i = 1$ if it prefers y_i to y_{ref} and $b_i = 0$ otherwise, giving the reward $R_i^{\text{rel}} = 0.1b_i$. Because relevance is an answer-level property, if the answer contains L_i tokens, each receives reward $0.1b_i/L_i$. We randomize whether the rollout is presented as

response A or B to reduce position bias and instruct the judge to assess only relevance and responsiveness, leaving factuality to the claim verifier. The prompt is provided in Listing 3.

Listing 3: Pairwise relevance-judging prompt.

```
You are given a user question and two AI-assistant responses (A and B). Judge which response more DIRECTLY and
      RELEVANTLY addresses the specific question the user asked.

Focus ONLY on relevance and responsiveness -- NOT on factual correctness (do not try to verify any facts; that is
       judged separately). Use these criteria:
- Directness: does the response answer what was actually asked, rather than a related-but-different question?
- On-topic focus: is the content about the user's specific question? A response padded with tangential, generic,
      or unrelated facts -- even if individually plausible -- is WORSE, not better.
- Appropriate scope: does it cover the key aspects the question calls for, without digressing into material the
      user did not ask about?

Do NOT reward a response for being longer or containing more facts. Judge the two answers on their merits
      regardless of the order in which they are presented.

[User Question]
{question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]

Provide your final verdict as ONLY a JSON object, with the reasoning FIRST and the winner LAST: {"reasoning":
      "...", "winner": "A"} if Assistant A is more relevant, or {"reasoning": "...", "winner": "B"} if Assistant
      B is more relevant.
```

B.5 Evaluation Trends and Statistical Significance

Table 3 reports the final step-100 GRPO, L-GRPO, and SDPO checkpoints and the best pre-collapse token-only checkpoint at step 30. To assess whether the main gains depend on checkpoint selection, we also evaluate the estimators every ten training steps. As shown in Figure 5, L-GRPO achieves higher factual precision and $\mathrm{R}_{\mathrm{fact}}$ at every evaluated checkpoint after training begins. It also produces more supported claims later in training and generally fewer not-supported claims, while reducing contradictions at every checkpoint after step 0. The sustained advantage across these complementary measures shows that the final results reflect a consistent training trend rather than a favorable checkpoint selected in isolation.

SDPO improves steadily across training but remains below both GRPO and L-GRPO on factual precision and $\mathrm{R}_{\mathrm{fact}}$ throughout the later checkpoints. Thus its lower final score does not result from selecting an unfavorable isolated checkpoint.

The token-only trajectory is qualitatively different. Its factual precision peaks at step 30 (53.3%), falls to 28.1% by step 40, and remains below 8% from step 60 onward. The mean number of supported claims falls concurrently from 14.8 at step 30 to approximately 2 per response. Table 3 therefore reports step 30 as an optimistic pre-collapse comparison; continued training substantially degrades the estimator's held-out performance.

Final-checkpoint significance. The GRPO and L-GRPO evaluations contain responses to the same 1,386 prompts. We estimate confidence intervals with 10,000 paired bootstrap resamples, sampling prompts within each benchmark and recomputing the unweighted macro-average over benchmark means. We compute two-sided p-values using 100,000 paired prompt-level randomizations. The macro-averaged factual-precision improvement is 6.3 percentage points (95% CI: 4.4–8.4; $p < 0.0001$). The secondary $\mathrm{R}_{\mathrm{fact}}$ improvement is 6.7 percentage points (95% CI: 4.0–9.5; $p < 0.0001$).

Table 8 reports the benchmark-level tests. We apply Bonferroni correction over the four benchmarks within each metric, giving a significance threshold of $p < 0.0125$. The factual-precision improvements on FactBench-Hard and Biography are significant under this correction. For $\mathrm{R}_{\mathrm{fact}}$, the improvements on LongFact, FactBench-Hard, and Factory-Hard reach corrected significance.

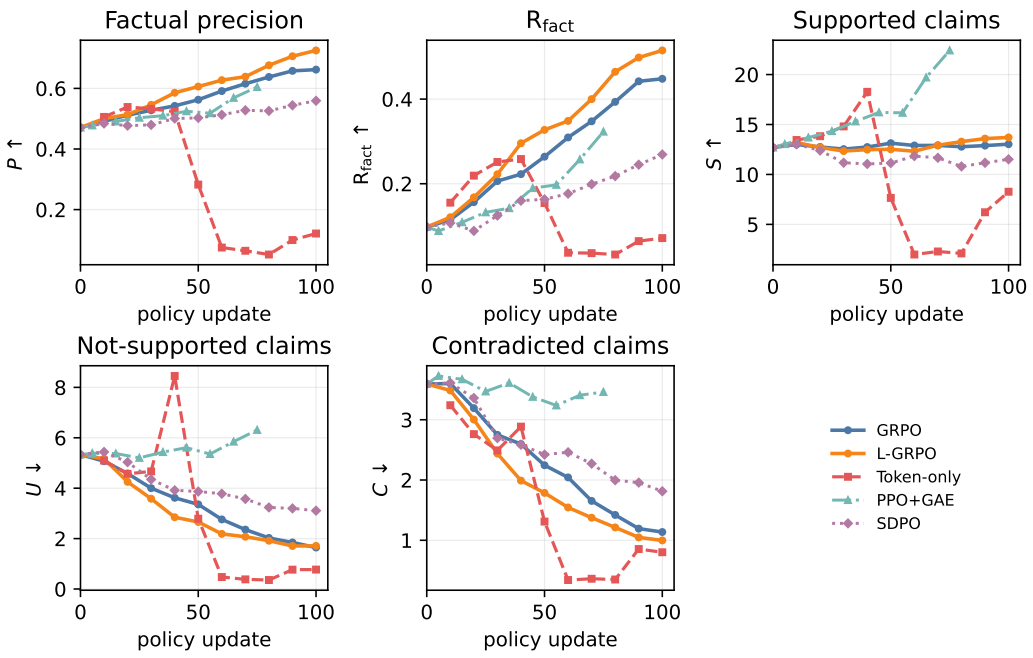


Figure 5: **Factuality evaluation over training.** Factual precision P, the reward-aligned $\mathrm{R}_{\mathrm{fact}}$, and supported ($S$), not-supported ($U$), and contradicted ($C$) claim counts. Each curve macro-averages the four benchmark means. L-GRPO outperforms GRPO on both summary metrics at every evaluated checkpoint after step 0. The token-only estimator improves initially before collapsing under continued training. SDPO improves steadily but remains below GRPO and L-GRPO; PPO uses $\lambda = 1$.

Table 8: **Factuality significance by benchmark.** Each cell reports the signed change from GRPO to L-GRPO followed by the paired-randomization p-value in parentheses. Changes in P and $\mathrm{R}_{\mathrm{fact}}$ are percentage points; changes in S, C, and U are claims per response. Positive changes favor L-GRPO for P, $\mathrm{R}_{\mathrm{fact}}$, and S, while negative changes favor it for C and U. * denotes a statistically significant change in either direction ($p < 0.0125$).

Metric	LongFact	FactBench-Hard	Factory-Hard	Biography	Macro-average
P	$+1.4$ ($.0738$)	$+2.4$ ($.0006^*$)	$+2.5$ ($.1136$)	$+19.0$ ($< .0001^*$)	$+6.3$ ($< .0001^*$)
$\mathrm{R}_{\mathrm{fact}}$	$+4.1$ ($.0017^*$)	$+3.5$ ($.0043^*$)	$+8.1$ ($.0003^*$)	$+11.1$ ($.0241$)	$+6.7$ ($< .0001^*$)
S	$+0.64$ ($.3106$)	-0.37 ($.3979$)	$+0.94$ ($.0011^*$)	$+1.49$ ($< .0001^*$)	$+0.68$ ($.0022^*$)
C	-0.31 ($.0003^*$)	-0.29 ($.0036^*$)	-0.19 ($.0086^*$)	$+0.24$ ($.0514$)	-0.14 ($.0039^*$)
U	$+0.11$ ($.2950$)	-0.45 ($.0004^*$)	$+0.46$ ($.0026^*$)	$+0.14$ ($.1912$)	$+0.06$ ($.2896$)

The macro-average weights each benchmark equally and is therefore driven substantially by Biography, which is small (183 prompts) but improves by 19.0 percentage points—roughly 75% of the macro-averaged gain; the other three benchmarks improve by 1.4–2.5 points. Weighting instead by prompt count, the micro-averaged factual precision rises from 68.6% to 73.0% (+4.4 points), a positive but smaller aggregate gain when the small Biography benchmark is not upweighted.

Claim-count diagnostics support the same interpretation. After correcting across the three count comparisons, the mean number of supported claims increases by 0.68 (95% CI: 0.24–1.11; $p = 0.0022$), while contradictions decrease by 0.14 claims per response (95% CI: 0.05–0.23 fewer; $p = 0.0039$). The change in not-supported claims is not significant ($p = 0.29$).

Table 9: **Per-benchmark factuality claim counts.** Mean supported, contradicted, not-supported, and not-enough-info claim counts ($S/C/U/I$) per response. Higher is better for S, while lower is better for C, U, and I; the best value for each count and benchmark is bold. All RL baselines use 12 rollouts per prompt. PPO uses $\lambda = 1$ and is reported at step 80, its last checkpoint before output-length degeneration.

Model	LongFact	FactBench-Hard	Factory-Hard	Biography
Qwen3-8B	21.8/3.4/4.6/0.7	17.1/3.0/4.7/2.5	6.3/2.1/5.9/5.1	5.5/5.9/6.1/3.3
RL w/ token-only	24.2/2.0/3.7/0.6	20.3/2.5/4.5/1.1	7.7/1.5/5.7/4.3	7.0/4.0/4.6/2.3
RL w/ PPO	**37.2**/4.1/7.4/0.7	**31.7**/4.5/7.4/1.8	**12.9**/2.0/7.5/4.6	**7.7**/3.3/3.0/0.7
RL w/ SDPO	17.6/1.4/2.0/0.3	17.4/2.1/3.6/1.2	6.4/1.7/4.9/4.3	4.6/2.1/2.0/0.6
RL w/ GRPO	22.9/1.1/**1.3**/0.2	18.6/1.6/2.7/0.6	6.3/1.0/**2.1**/**1.5**	4.3/**0.9**/**0.6**/**0.2**
RL w/ L-GRPO	23.5/**0.8**/1.4/**0.2**	18.2/**1.3**/**2.2**/**0.5**	7.3/**0.8**/2.5/1.9	5.8/1.1/0.7/**0.2**

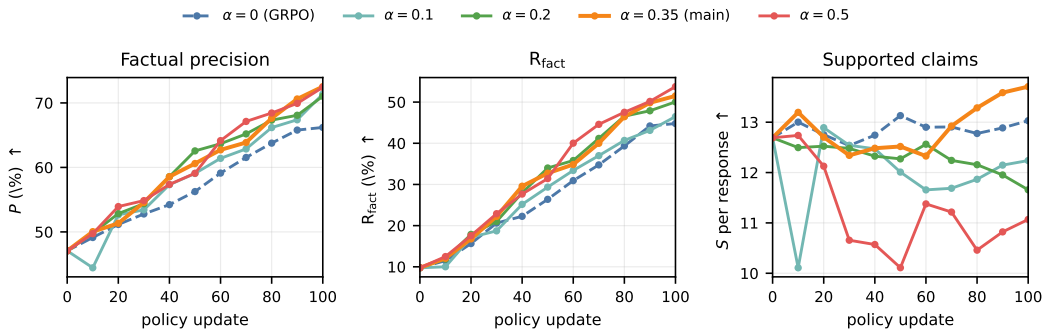


Figure 6: **Sensitivity to the localization weight on factuality.** Macro-averaged factual precision, reward-aligned $\mathrm{R}_{\mathrm{fact}}$, and supported claims per response over training. Positive localization weights converge to similar factual precision, while larger values increasingly favor the reward-aligned metric over supported-claim coverage.

B.6 Sensitivity to the Localization Weight

We rerun factuality training with $\alpha \in \{0.1, 0.2, 0.5\}$, holding the remaining settings fixed, and compare against GRPO ($\alpha = 0$) and the main $\alpha = 0.35$ run. Figure 6 shows the complete evaluation trajectories. At step 100, every positive value improves factual precision over $\alpha = 0$ (71.0–72.5% versus 66.2%). Averaged over the final three evaluations, factual precision is 68.3%, 68.8%, 70.3%, and 70.3% for $\alpha = 0.1$, 0.2, 0.35, and 0.5, respectively, compared with 65.3% for $\alpha = 0$. The benefit therefore does not depend on a narrowly tuned positive weight.

The reward-aligned $\mathrm{R}_{\mathrm{fact}}$ increases more consistently with α, but larger weights also favor shorter, more conservative responses. Over the same final window, $\alpha = 0.5$ produces 20% fewer supported claims and 24% fewer sentences than $\alpha = 0.35$, despite the two settings having identical factual precision to one decimal place. We therefore use $\alpha = 0.35$ as a balance between factual precision and coverage.

C Training and Implementation Details

C.1 Span→token matching pseudocode

LLM judges identify rewarded spans as verbatim substrings of the response. To assign each span's reward to the corresponding generated tokens, we map it to the policy model's tokenization of the response. Algorithm 2 gives the matching heuristic.

```
Algorithm 2: Span-to-token matching.
Input: judged span s; answer text y; answer token IDs z;
       policy tokenizer tau
Output: matched token indices T, or UNMAPPED(reason)

y = normalize_whitespace(y)
s = normalize_whitespace(s)

if s == "":
    return UNMAPPED("empty")
if s not in y:
    return UNMAPPED("zero_match")

encoded = []
for variant in [s, " " + s]:
    ids = tau.encode(variant)
    if ids is not empty:
        encoded.append(ids)

if encoded is empty:
    return UNMAPPED("encode_fail")

matches = []
for ids in encoded:
    local = exact_subsequence_matches(ids, z)
    if local is empty and len(ids) >= 4:
        local = endpoint_tolerant_matches(ids, z)
    matches += local

matches = unique(matches)
if len(matches) == 1:
    return matches[0]
if len(matches) == 0:
    return UNMAPPED("zero_match")
return UNMAPPED("multi_match")
```

Here `exact_subsequence_matches` returns every position where `ids` occurs as a contiguous subsequence of the answer token IDs `z`. When that fails for a span of at least four tokens, `endpoint_tolerant_matches` relaxes the two boundary tokens: it matches the interior subsequence `ids[1:-1]` and accepts a candidate when only the first and/or last token differs, absorbing tokenizer boundary effects (such as leading-whitespace merges) that can make a span tokenize differently in isolation than in context. The exact procedure is included in our released code.

The strings returned by `UNMAPPED` are diagnostic categories for empty judge spans, tokenizer failures, spans with no token match, and spans with multiple candidate matches. Across MT and factuality, adopting minimal verbatim spans and unique-match selection reduced multi-match failures from $\sim$64% to $< 1\%$ and successfully localized $\sim$98% of judged spans.

C.2 Graded Tokens for Localized Redistribution

The localized term of L-GRPO (Eq. 7) redistributes advantage only among the tokens the judge grades. Let G_i be the graded-token set for rollout i. We set

$$\bar{r}_i = \frac{1}{|G_i|} \sum_{t \in G_i} r_{i,t}, \qquad d_{i,t} = \begin{cases} r_{i,t} - \bar{r}_i & t \in G_i, \\ 0 & t \notin G_i, \end{cases}$$

and pool σ_d over $\bigcup_i G_i$. Because $\sum_{t \in G_i} d_{i,t} = 0$, the redistribution stays mean-zero over the whole rollout, so Eq. 8 is unchanged and every ungraded token receives exactly $\hat{A}_i^{\mathrm{GRPO}}$. The graded set is task-specific:

- **Machine translation:** G_i is the set of translation (answer) tokens. The judge evaluates the entire translation, so all answer tokens are graded—marked error spans receive nonzero $r_{i,t}$ and unmarked tokens are assessed as correct with $r_{i,t}$=0; only chain-of-thought tokens are excluded.
- **Long-form factuality:** G_i is the set of tokens lying within an extracted claim span. Chain-of-thought tokens and answer tokens outside any claim (e.g., connective or discourse text) are excluded, since the verifier assigns them no verdict.

Restricting redistribution to G_i keeps ungraded tokens governed purely by the rollout-level signal $\hat{A}_i^{\mathrm{GRPO}}$: the localized term neither rewards nor penalizes them.

Table 10: **Training and generation hyperparameters.** Values shared by GRPO and L-GRPO are listed once; the two values of α identify the respective estimators.

Hyperparameter	MT	Factuality
Optimization		
Policy model	Qwen3-8B	Qwen3-8B
Parameter update	Full	Full
Training precision	bfloat16	bfloat16
Training steps	150	100
Checkpoint reported in results	150	100
Prompts per update	64	64
Rollouts per prompt	12	12
Rollouts per update	768	768
Optimizer updates per rollout batch	1	1
Actor micro-batch size per GPU	1	2
Learning rate	1×10^{-6}	2×10^{-6}
Learning-rate schedule	Constant	Constant
Warmup steps	0	0
Weight decay	0.01	0.01
Gradient clipping	1.0	1.0
PPO clipping ratio	0.2	0.2
Reference-policy KL estimator	k_3 (low variance)	k_3 (low variance)
KL coefficient	1×10^{-3}	1×10^{-3}
Entropy coefficient	0	0
Loss aggregation	Token mean	Token mean
GRPO advantage standardization	Yes	Yes
α (GRPO / L-GRPO)	0 / 0.1	0 / 0.35
Checkpoint/evaluation interval	10 steps	10 steps
Training generation		
Maximum prompt length	2,048	2,048
Maximum response length	8,192	8,192
Temperature	1.0	1.0
Top-p	1.0	1.0
Top-k	Disabled	Disabled
Evaluation generation		
Samples per prompt	1	1
Maximum response length	8,192	16,384
Decoding	Greedy	Sampling
Temperature	0	1.0
Top-p	1.0	1.0
Top-k	Disabled	Disabled
Distributed execution		
GPUs per run	8	8
Rollout backend	vLLM	vLLM
Tensor parallel size	2	2
Training strategy	FSDP2	FSDP2

C.3 Training Hyperparameters

Table 10 lists the optimization and generation settings used for the reported MT and factuality experiments. GRPO and L-GRPO use identical settings within each task except for the localization weight α.

C.4 PPO Baseline Hyperparameters

The PPO baseline uses the same Qwen3-8B policy, training data, reward construction, prompt batch size, and 12 rollouts per prompt as the corresponding GRPO and L-GRPO runs. It additionally trains a separate Qwen3-8B value network and uses GAE to convert the token rewards into token-

Table 11: **PPO baseline hyperparameters.** Slash-separated entries list policy/value-network settings, respectively.

Hyperparameter	MT	Factuality
Advantage estimator	GAE	GAE
Discount factor γ	1.0	1.0
GAE parameter λ	1.0	1.0
Value-network target	Monte Carlo return	Monte Carlo return
Policy/value learning rate	1×10^{-6} / 1×10^{-5}	2×10^{-6} / 1×10^{-5}
Policy/value optimization epochs	1 / 1	1 / 1
Policy/value mini-batch size	64 / 16	64 / 16
Policy/value micro-batch size per GPU	1 / 1	1 / 1
Policy clipping ratio	0.2	0.2
Value clipping ratio	0.5	0.2
Reference-policy KL estimator	k_3 (low variance)	k_3 (low variance)
KL coefficient	1×10^{-3}	1×10^{-3}
Critic-only warm-up updates	5	5
Maximum response length	8,192	8,192
Checkpoint reported in results	150	80

level advantages. Table 11 lists the PPO-specific settings. We swept the GAE parameter $\lambda \in \{0.95, 0.99, 1.0\}$ and found $\lambda = 1.0$ to perform best, which we use throughout; that PPO does best when relying least on the value network for temporal-difference bootstrapping is consistent with the value-estimation limitations documented in App. C.6.

C.5 SDPO Baseline Hyperparameters

We implement SDPO using the same policy, data, judge outputs, and rollout-group size as GRPO. Following Eq. 4, the clipped GRPO loss is combined with feedback-conditioned symmetric JSD using $\beta = 0.2$. Symmetric JSD is the stability variant recommended in the original SDPO work. We tuned rollout sampling (temperature, top-p, top-k) separately for each baseline and report the best-performing configuration; PPO and SDPO both favored more deterministic sampling than GRPO and L-GRPO, which for SDPO reduces sampling noise in the self-distillation objective.

The self-teacher re-prompt is formed by appending a task-specific rendering of the same judge output used to construct rewards to the already-templated original user turn, immediately before the assistant-generation header. The sampled response is not repeated in the augmented user turn. For MT, the feedback contains the severity, category, quoted span or omission, and note for each MQM item. For factuality, it contains contradicted and not-supported claims with evidence and supported claims without evidence; not-enough-info claims are excluded. Listings 4 and 5 show the fixed templates used in the final runs; braces denote instance-specific fields and repeated items.

Listing 4: MT self-teacher re-prompt.

```
{original translation request and source text}

An expert evaluator reviewed your previous attempt at translating the source above. It was
asked to find EVERY error and return each one as an exact quoted span. "major" means the error
substantially changes meaning or harms understanding, "minor" means it does not; the note says
what is wrong and why. That attempt is not shown here.

Errors the evaluator found in your previous attempt:
- [{severity}/{category}] "{target span}" — {note}
- [{severity}/omission] MISSING from your translation: "{source span}" — {note}

Now produce a better translation of the original source, avoiding the errors listed above.
```

Listing 5: Factuality self-teacher re-prompt.

```
{original user question}

An automated fact-checker reviewed your previous answer to this question. It split that answer
into individual factual claims and searched on the web for sources on each one. Its findings
are grouped below by what the sources showed.
```

Table 12: **SDPO baseline hyperparameters.** Settings for the reported baseline.

Hyperparameter	MT	Factuality
GRPO loss retained	Yes	Yes
β (distillation loss weight)	0.2	0.2
Teacher conditioning	Judge feedback	Judge feedback
Successful sibling	None	None
Teacher-selected support size k	20	20
JSD mixture parameter	0.5	0.5
Teacher interpolation coefficient η	0.1	0.1
Tail probability bucket	Enabled	Enabled
Reference-policy KL coefficient	10^{-3}	10^{-3}
Actor micro-batch size per GPU	1	1
Sampling/JSD temperature	0.6	0.6
Rollout top-p	0.95	0.95
Rollout top-k	20	-1 (disabled)
Checkpoint reported in results	100	100

```
CONTRADICTED by the sources -- these are wrong:
  1. Claim: {contradicted claim}
     Evidence: {checker evidence}

UNVERIFIED -- the search found no evidence either way on these:
  1. Claim: {not-supported claim}
     Evidence: {checker evidence}

CONFIRMED by the sources:
  1. Claim: {supported claim}

Write a better answer, which is more factually correct and relevant to the user's question. Keep the confirmed
      claims and anything else you are confident is correct. Do not include any of the contradicted claims --
      those are wrong. A claim with no supporting source is not necessarily wrong; the checker simply found no
      evidence either way. Keep such a claim if you are confident it is correct, but do not present it as
      established fact -- and leave it out if you are not confident. Apply the same judgement to anything else
      you are unsure of.
```

We also experimented with supplying the highest-reward alternative rollout in the group whenever its reward exceeded that of the current rollout by a preset margin. This yielded worse results, so the final MT and factuality runs do not include a sibling response.

Following SDPO, we form a trust-region teacher by geometrically interpolating the frozen-reference and current feedback-conditioned distributions:

$$\widetilde{q}_{i,t}(v) \propto \exp\big((1-\eta)\log q^{\mathrm{ref}}_{i,t}(v) + \eta \log q^{\theta}_{i,t}(v)\big)\,, \tag{9}$$

where η is the teacher interpolation coefficient, and q^{ref} and q^{θ} are respectively the frozen-reference and current policy distributions under the feedback-conditioned prompt. The teacher does not sample a new response; it teacher-forces the original response. Nevertheless, the student and teacher distributions used in the JSD are both evaluated at temperature 0.6, so temperature affects their probabilities.

For efficiency, we compute symmetric JSD over a shared top-20 support plus a tail bucket. All teacher scores are stop-gradient targets computed before the actor update, and the loss is averaged over response positions.

C.6 PPO Critic Diagnostics

We examine how accurately the PPO value network predicts the sampled reward-to-go used to construct its token-level advantages. Because our PPO runs use $\gamma = 1$, the Monte Carlo reward-to-go from token t is $G_{i,t} = \sum_{l=t}^{T_i} r_{i,l}$. We first compute

$$\mathrm{EV}_{\mathrm{MC}} = 1 - \frac{\mathrm{Var}(G_{i,t} - V_\psi(s_{i,t}))}{\mathrm{Var}(G_{i,t})}, \tag{10}$$

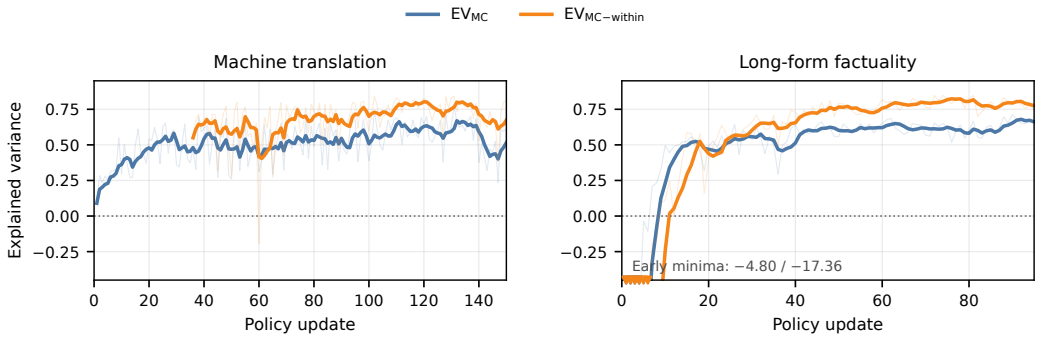


Figure 7: **PPO value-network diagnostics over training.** $\mathrm{EV}_{\mathrm{MC}}$ measures prediction of the remaining rollout reward, while $\mathrm{EV}_{\text{MC-within}}$ removes each response's mean and measures discrimination among token positions within that response. Faint lines show individual updates and solid lines show five-update rolling means. Higher is better, and zero denotes the corresponding constant-mean baseline. The earliest factuality values below the displayed range are marked at the lower boundary. MT's within-response diagnostic was logged only after policy update 35.

over all valid generated tokens in a training batch. A value of one denotes perfect prediction, zero is no better than predicting the batch mean, and a negative value is worse than that constant baseline.

This aggregate measure can be high if the critic distinguishes generally high-reward responses from low-reward responses without identifying how the remaining reward changes across token positions within a response. We therefore also report $\mathrm{EV}_{\text{MC-within}}$. For each response, we subtract the response-level token mean from both $G_{i,t}$ and $V_\psi(s_{i,t})$, then apply the same explained-variance calculation. This removes between-response differences and measures whether the critic distinguishes token positions within an individual response. Figure 7 shows both diagnostics over training.

The factuality critic therefore begins with little ability to predict either the remaining rollout reward or where that reward changes within a response. It improves rapidly and captures substantial reward structure later in training, but its predictions remain imperfect. The MT critic likewise explains a meaningful but incomplete fraction of both overall and within-response variation. These diagnostics support the hypothesis that PPO's learned baseline introduces estimation error during the conversion from localized rewards to token-level advantages.

C.7 Training Dynamics Comparison

Figure 8 compares the training dynamics of GRPO, L-GRPO, PPO, SDPO, and the token-only estimator on both tasks. GRPO and L-GRPO remain stable throughout training: their rewards improve steadily, with L-GRPO maintaining a higher reward over most of MT training and a smaller advantage over much of factuality training. Relative to GRPO, L-GRPO produces a larger but controlled policy shift, reflected in higher KL and longer chain-of-thought. Its entropy remains bounded, and neither task exhibits the reward collapse seen with token-only estimation.

SDPO also avoids an outright reward collapse, but it induces a markedly larger policy shift than GRPO or L-GRPO. Its KL rises substantially on both tasks, while its validation chain-of-thought grows rapidly for MT and contracts for factuality. Together with its weaker held-out results, these dynamics suggest that the feedback-conditioned distillation loss changes the policy without converting the judge's localized errors into equally effective task-level improvements.

The token-only estimator initially tracks the stable estimators but then diverges. In MT, its reward collapses after step 80 as KL rises sharply and entropy falls; its validation chain-of-thought also becomes shorter. In factuality, the failure begins earlier, around step 40: reward approaches zero, KL spikes, entropy collapses, and mean validation chain-of-thought length grows toward the response limit. The opposite length trends across the two tasks indicate a general optimization failure rather than a consistent preference for shorter or longer responses.

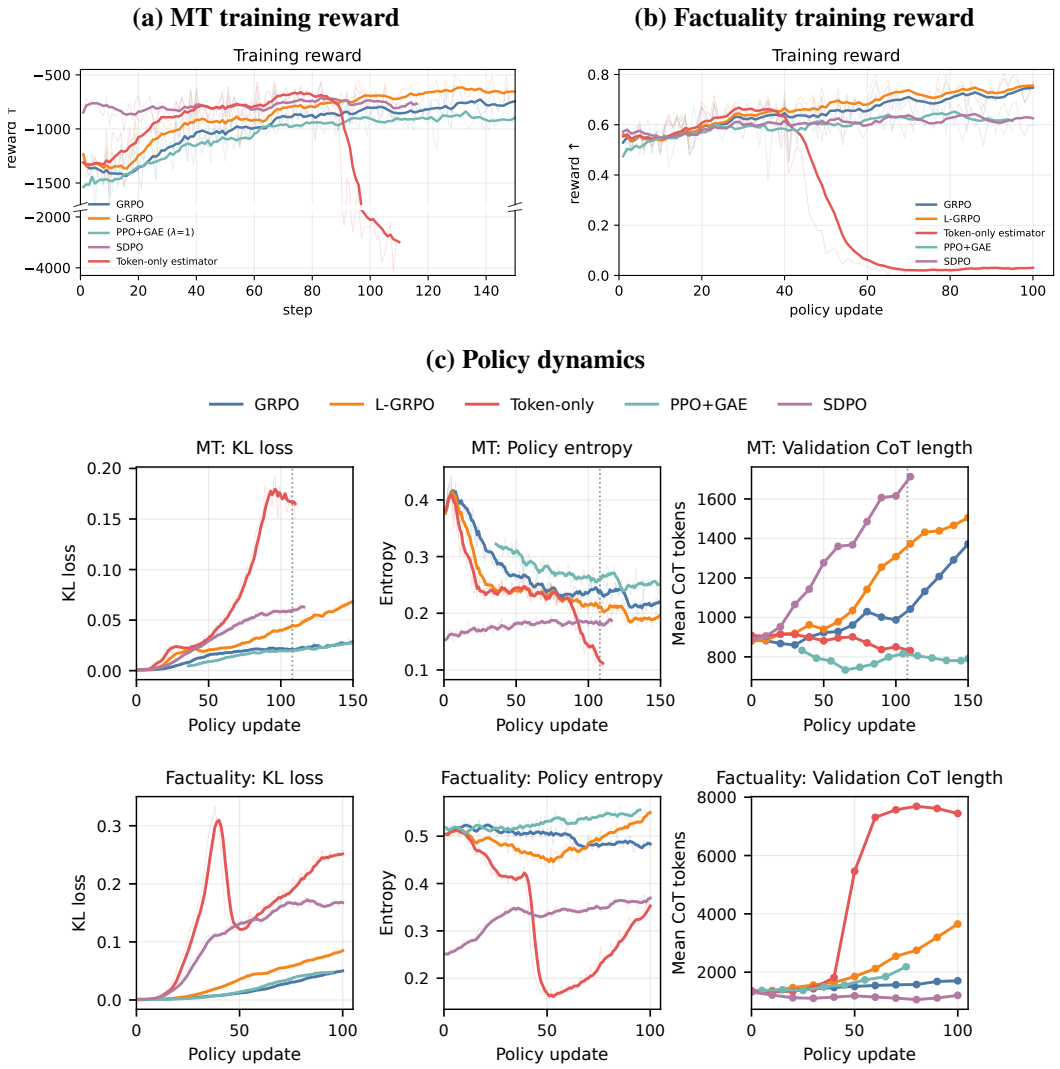


Figure 8: **Training dynamics across advantage estimators.** (a–b) Mean training reward for MT and factuality; higher is better. The MT panel uses a broken y-axis to retain the scale of the stable runs, while the factuality panel uses a continuous axis. (c) Training KL loss, policy entropy, and mean validation chain-of-thought length for MT (top) and factuality (bottom). Faint lines are raw training steps, solid lines are rolling means, and the dotted line marks the step-108 token-only MT training failure. PPO uses $\lambda = 1$.

C.8 Length Penalty and Overlong Masking

We use the same response-length controls for GRPO and L-GRPO within each task. Let ℓ_i denote the number of generated tokens in rollout i. Before group normalization, we subtract the soft length penalty

$$p_{\text{len}}(\ell_i) = \lambda_{\text{len}} \left[\text{clip}\left(\frac{\ell_i - L_1}{L_2 - L_1}, 0, 1 \right) \right]^q$$

from the rollout return. We set L_1=6000, L_2=8192, and q=1 for both tasks, with λ_{len}=2000 for MT and λ_{len}=0.05 for factuality; the different scales reflect the corresponding reward ranges. Thus the penalty is zero through 6,000 tokens and increases linearly toward the generation cap.

Rollouts with $\ell_i \geq 8000$ are excluded when computing their group's return mean and standard deviation, and their token advantages are set to zero. This prevents a small number of truncated

responses with extreme rewards from distorting the group baseline or contributing a policy update. The soft penalty and mask modify only the rollout-level term and are applied identically to GRPO and L-GRPO.