Edit-Aware Reward Modeling for Chinese Grammatical Error Correction

Yilin Li, Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
liyilin25@stu.pku.edu.cn, wanxiaojun@pku.edu.cn

Abstract

While large language models have achieved remarkable success in various natural language processing tasks, their potential in grammatical error correction remains underexplored. Recent work has applied reinforcement learning with rule-based rewards to CGEC, but these approaches rely on coarse-grained binary signals (exact match or not) that fail to capture fine-grained quality distinctions among correction candidates. In this paper, we propose **Edit-Aware Reward Model (EARM)**, a novel reward modeling framework that explicitly incorporates edit-awareness into preference learning for CGEC. EARM introduces a dual-granularity training objective that jointly optimizes sentence-level and token-level weighted Bradley-Terry ranking losses, where edit tokens receive higher importance weights. When integrated with GRPO, our approach achieves 61.29/63.08 $F_{0.5}$ on FCGEC/NaCGEC (single output), and 65.04/64.59 with best-of-16 reranking, surpassing previous best by 5.41 and 1.80 points. Extensive experiments demonstrate that learned edit-aware rewards significantly outperform rule-based alternatives for CGEC preference optimization.

1 Introduction

Grammatical Error Correction (GEC) is a fundamental task in Natural Language Processing (NLP), which aims to automatically detect and correct grammatical errors in text with minimal edits (Bryant et al., 2023). GEC models have widespread applications in writing assistants, language learning systems, and search engines. Traditional GEC methods can be divided into Sequence-to-Edit (Seq2Edit) approaches that predict edit operations for each token (Omelianchuk et al., 2020), and Sequence-to-Sequence (Seq2Seq) approaches that treat GEC as monolingual translation (Zhang et al., 2022b).

Recently, decoder-only large language models (LLMs) have demonstrated remarkable capabilities across various NLP tasks. However, initial studies suggest that LLMs struggle to surpass traditional models on GEC (Fang et al., 2023; Qu et al., 2025), primarily due to the conflict between their generative nature and the minimal edit principle of GEC. LLMs often lead to overcorrection, where grammatically correct segments are unnecessarily modified(Omelianchuk et al., 2024).

Candidate	Rule	EARM
Limitation 1: Coarse Granularity *Cannot distinguish partial, unchanged, and overcorrection*		
Source: 不但他学习好，而且身体也不好。 **Reference:** 他不但学习好，而且身体也好。		
他不但学习好，而且身体也好。 (exact)	1.0	2.30
不但他学习好，而且身体也好。 (partial)	0.0	1.48
不但他学习好，而且身体也不好。 (unchanged)	0.0	-0.86
他学习成绩优秀，身体素质也很棒。 (overcorr.)	0.0	-6.47
Limitation 2: Score Collision *Valid alternative and error get same reward*		
Source: 他对学习数学很感兴趣极了。 **Reference:** 他对学习数学很感兴趣。		
他对学习数学很感兴趣。 (exact)	1.0	3.50
他对学习数学感兴趣极了。 (valid alt.)	0.0	3.63
他对数学很感兴趣。 (new error)	0.0	-4.59

Table 1: Limitations of rule-based rewards and comparison with EARM rewards. Rule-based rewards exhibit coarse granularity and score collision, while EARM rewards provide fine-grained distinctions that address both issues.

Reinforcement learning (RL) offers a promising direction to align LLMs with task-specific requirements. Recent work has applied Group Relative Policy Optimization with rule-based rewards to GEC, achieving notable improvements (Li et al., 2025). However, rule-based rewards suffer from two fundamental limitations, as illustrated in Table 1:

Limitation 1: Coarse Granularity. Rule-based

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 40945–40957

rewards provide only binary signals based on exact string matching—a candidate either matches the reference exactly (reward=1) or not (reward=0). This fails to distinguish between partial corrections, unchanged inputs, and complete rewrites: a candidate that fixes one of two errors receives the same zero reward as one that makes no correction or overcorrection.

Limitation 2: Score Collision. GEC often admits multiple valid corrections, but references may not cover all acceptable answers. Acceptable alternatives using synonyms or different correction strategies receive zero reward, identical to genuinely incorrect outputs or overcorrections.

To address these limitations of rule-based reward which may provide uninformative gradients for policy learning, we propose **Edit-Aware Reward Model (EARM)**, a learned reward model that explicitly captures the editing structure of GEC outputs. Our key insight is that **not all tokens contribute equally to correction quality**—tokens at edit boundaries (pivot tokens) and within edited spans carry more discriminative information than unchanged tokens. As illustrated in Figure 1, EARM assigns differentiated weights to tokens based on their editing roles during reward computation.

Specifically, EARM employs character-level alignment to identify edit operations and assigns differentiated importance weights: pivot tokens, continuation tokens and unchanged tokens receive different weights, respectively. The model is trained with a dual-granularity objective combining sentence-level and token-level weighted ranking losses, where the token-level loss guides the backbone to learn edit-aware representations. Unlike rule-based binary rewards or offline methods DPO, EARM provides continuous, fine-grained signals for effective online policy optimization.

When integrated with GRPO, our approach achieves 61.29/63.08 $F_{0.5}$ on FCGEC/NaCGEC (single output), and 65.04/64.59 with best-of-16 reranking.

Our contributions are:

- We propose edit-aware token weighting to address the granularity mismatch in rule-based GEC rewards.
- We design EARM with a dual-granularity training objective that jointly optimizes sentence-level and token-level weighted ranking losses.
- We achieve state-of-the-art results on widely-used CGEC benchmarks, demonstrating that learned edit-aware rewards significantly outperform rule-based alternatives for preference optimization[1].

2 Related Work

2.1 Grammatical Error Correction

Traditional GEC methods fall into two categories. **Seq2Edit** approaches frame GEC as sequence labeling, predicting edit operations for each token. GECToR (Omelianchuk et al., 2020) introduces task-specific token transformations, achieving high-precision results with efficient inference. **Seq2Seq** approaches treat GEC as monolingual translation using encoder-decoder architectures. Models based on BART (Lewis et al., 2019) and T5 (Raffel et al., 2019) have achieved strong results, with SynGEC (Zhang et al., 2022b) further incorporating syntactic information.

While LLMs have achieved success in many NLP tasks, they face challenges in GEC due to the conflict between their generative nature and the minimal edit principle (Fang et al., 2023; Coyne et al., 2023). Several studies apply supervised fine-tuning to enhance LLM performance on GEC (Zhang et al., 2023), while others improve closed-source models through prompt engineering (Coyne et al., 2023). However, these methods have yet to achieve the impressive performance seen in other NLP tasks.

2.2 Preference Learning for LLMs

LLM alignment has evolved from reinforcement learning from human feedback (RLHF) (Kaufmann et al., 2023) and proximal policy optimization (PPO) (Schulman et al., 2017) to more efficient offline methods. Direct Preference Optimization (DPO) (Rafailov et al., 2023) eliminates explicit reward models by directly optimizing on preference pairs, with variants like SimPO (Meng et al., 2024) further simplifying the training process. EPO (Liang et al., 2025) is a DPO variant for GEC. However, offline methods suffer from distribution shift and operate without explicit reward models, limiting their applicability to online RL.

Online methods like Group Relative Policy Optimization (GRPO) (Shao et al., 2024) address this

[1]Our trained reward model and code are available at https://github.com/GoldenLinlin/EARM.

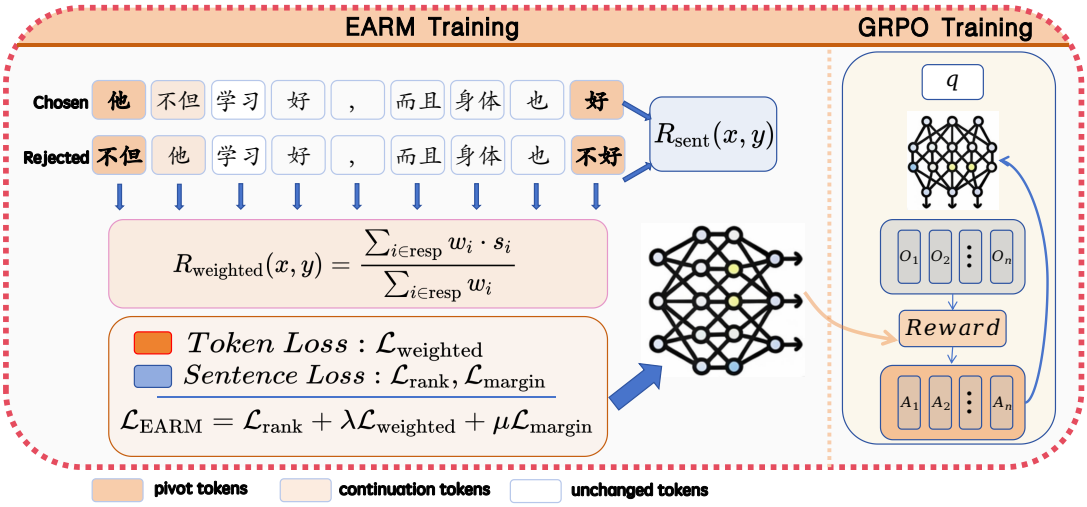

Figure 1: Overview of our Edit-Aware Reward Model (EARM) framework. Given a source sentence and correction candidate, EARM first identifies edit spans through character-level alignment, then computes edit-aware token weights. The model outputs both sentence-level rewards and token-level weighted scores, which are used to train the reward model with a hybrid loss and subsequently guide GRPO policy optimization. Here x denotes the source sentence, and y denotes the chosen or rejected candidate.

by enabling dynamic policy updates with group-based advantage estimation. DeepSeek-R1 (Guo et al., 2025) demonstrated that large-scale RL with simple rule-based rewards can elicit emergent reasoning capabilities. Recent work (Li et al., 2025) applies GRPO to GEC with heuristic rewards based on exact string matching, achieving notable improvements. However, binary rewards fail to capture fine-grained quality distinctions among correction candidates, motivating the need for learned reward models.

2.3 Reward Modeling

Reward modeling has become central to aligning LLMs with human preferences. Bradley-Terry models (Bradley and Terry, 1952) form the foundation of most approaches, learning to predict preferences from pairwise comparisons. Recent work explores token-level rewards (Zhong et al., 2024) and process rewards (Lightman et al., 2024) for improved credit assignment. However, these approaches are designed for general text generation and do not account for the specific characteristics of editing tasks like GEC.

Our work bridges preference optimization and reward modeling for GEC by learning an edit-aware reward model that provides continuous, fine-grained signals for policy optimization.

3 Methodology

3.1 Problem Formulation

Given a source sentence x with grammatical errors and a set of candidate corrections $\{y_1, y_2, ..., y_n\}$, our goal is to learn a reward model $R(x, y)$ that accurately ranks candidates by correction quality. Unlike rule-based rewards that provide binary signals, we aim to learn continuous rewards that capture subtle quality differences:

$$R(x, y_i) > R(x, y_j) \Leftrightarrow \text{quality}(y_i) > \text{quality}(y_j) \quad (1)$$

3.2 Edit-Aware Token Weighting

The core innovation of EARM is explicit modeling of edit operations at the token level. Given source x and candidate y, we perform character-level alignment using ChERRANT (Zhang et al., 2022a) for Chinese to identify edit spans.

Edit Mask Construction We construct an edit mask $M \in \{0, 1, 2\}^{|y|}$ for each token in the candidate:

- $M_i = 0$: Token i is unchanged from source
- $M_i = 1$: Token i is a **pivot token** (edit boundary)
- $M_i = 2$: Token i is a **continuation token** (within edit span)

Weight Assignment We convert the edit mask to importance weights:

$$w_i = \begin{cases} \alpha & \text{if } M_i = 1 \text{ (pivot)} \\ \gamma & \text{if } M_i = 2 \text{ (continuation)} \\ 1 & \text{if } M_i = 0 \text{ (unchanged)} \end{cases} \quad (2)$$

where $\alpha > \gamma > 1$. The intuition is that pivot tokens mark where corrections begin and carry the most discriminative information about correction decisions. Continuation tokens within edit spans are important but secondary to boundaries.

Edit Mask Examples The pivot and continuation token labels are derived from the character-level alignment output of ChERRANT. For each edit operation, ChERRANT produces an aligned span between the source and the candidate. We illustrate the three edit types:

- **Substitution** (e.g., "削弱" → "防止"): The first target token "防" is the pivot token ($M_i = 1$), and "止" is the continuation token ($M_i = 2$).
- **Insertion** (e.g., inserting "的" between two tokens): The inserted token "的" is the pivot token ($M_i = 1$). Since the edit span length is 1, there are no continuation tokens.
- **Deletion** (e.g., removing "了" from the source): The target side has no corresponding token for the deleted character. In this case, the token immediately following the deletion point is marked as pivot ($M_i = 1$), signaling that an edit decision occurred at this boundary.

In all cases, the pivot token marks the position where an edit decision occurs in the response, serving as the most informative position for the reward model.

3.3 Model Architecture

EARM builds upon a pre-trained language model backbone with two prediction heads.

Value Head To obtain the sentence-level scores, we apply a two-layer MLP over the hidden state corresponding to the last token.

$$R_{\text{sent}}(x, y) = \text{MLP}_{\text{value}}(h_{[\text{EOS}]}) \quad (3)$$

Token Score Head A two-layer MLP produces per-token scores:

$$s_i = \text{MLP}_{\text{token}}(h_i) \quad (4)$$

Weighted Token Score Aggregation We compute the edit-aware weighted score by aggregating token scores with edit weights, restricted to the response region:

$$R_{\text{weighted}}(x, y) = \frac{\sum_{i \in \text{resp}} w_i \cdot s_i}{\sum_{i \in \text{resp}} w_i} \quad (5)$$

Why Two Heads? We use sentence-level rewards R_{sent} for GRPO training, as this aligns naturally with its advantage computation mechanism, which estimates relative quality across candidates within a group—token-level rewards would require non-trivial modifications to compute per-token advantages and are left for future exploration. While only R_{sent} is used during GRPO, the token score head plays a crucial role during reward model training. Both heads share the same backbone, and the token-level weighted loss $\mathcal{L}$weighted (Equation 7) explicitly encourages the backbone to learn edit-aware representations. These improved representations benefit the value head, resulting in sentence-level rewards that better capture edit-relevant quality distinctions. Ablation studies confirm this design: removing $\mathcal{L}$weighted degrades the reward model's ranking accuracy (Table 3).

3.4 Training Data Construction

We construct preference pairs from the FCGEC dataset, using its training set for reward model training and validation set for development. The construction pipeline proceeds as follows:

1. **Candidate Generation**: For each source sentence, we use the SFT model (described in Section 3.6) to generate 8 correction candidates via high-temperature sampling ($T = 1.0$), encouraging diverse outputs that span a range of correction qualities.
2. **Quality Scoring**: We define the quality metric $\text{dist}(y, \text{ref})$ as the character-level edit distance between candidate y and the reference correction, computed using ChERRANT alignment. Lower distance indicates higher quality.
3. **Pair Selection**: We construct preference pairs (y_w, y_l) where $\text{dist}(y_w, \text{ref}) < \text{dist}(y_l, \text{ref})$, ensuring the chosen candidate y_w is closer to the reference than the rejected candidate y_l. For each source sentence, we randomly sample up to k pairs ($k = 10$ for training, $k = 3$

for validation), with at least one pair containing a reference-matching candidate when available.

3.5 Training Objective

We train EARM with a hybrid loss combining three components.

Sentence-Level Ranking Loss Standard Bradley-Terry loss on sentence-level reward:

$$\mathcal{L}_{\text{rank}} = -\log \sigma(R_{\text{sent}}(x, y_w) - R_{\text{sent}}(x, y_l)) \quad (6)$$

where y_w and y_l are chosen and rejected candidates, and σ is the sigmoid function.

Weighted Token-level Loss Bradley-Terry loss on edit-aware weighted scores:

$$\mathcal{L}_{\text{weighted}} = -\log \sigma(R_{\text{weighted}}(x, y_w) - R_{\text{weighted}}(x, y_l)) \quad (7)$$

Margin Loss Explicit margin constraint for robust separation:

$$\mathcal{L}_{\text{margin}} = \max(0, m - (R_{\text{sent}}(x, y_w) - R_{\text{sent}}(x, y_l))) \quad (8)$$

Combined Objective

$$\mathcal{L}_{\text{EARM}} = \mathcal{L}_{\text{rank}} + \lambda \mathcal{L}_{\text{weighted}} + \mu \mathcal{L}_{\text{margin}} \quad (9)$$

where λ and μ control the contribution of each component.

3.6 Integration with GRPO

After training EARM, we optimize a GEC policy model through SFT warm-up followed by RL. Following Li et al. (2025), we perform two-stage supervised fine-tuning: stage 1 combines Lang8 and HSK (replicated 5×) with reasoning traces generated by Qwen3-32B, yielding 1.57M pairs; stage 2 fine-tunes on FCGEC with reasoning traces generated by DeepSeek-R1 and filtered by DeepSeek-V3, obtaining 27.5K high-quality instances.

The SFT model then initializes GRPO training (Shao et al., 2024) with EARM as the reward function. For each source q, we sample candidates $\{o_1, ..., o_G\}$ and compute advantages:

$$A_i = \frac{R_{\text{EARM}}(q, o_i) - \text{mean}(\{R_j\})}{\text{std}(\{R_j\})} \quad (10)$$

GRPO then optimizes the policy π_θ:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q),\, \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \min\Big(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)} A_i, \text{clip}\Big(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)}, 1 - \varepsilon_{low}, 1 + \varepsilon_{high} \Big) A_i \Big) - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \Bigg] \quad (11)$$

where ε_{low}, ε_{high} control the clipping threshold, and β regulates the KL divergence penalty to prevent excessive policy shifts from the reference policy π_{ref}.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate on FCGEC (Xu et al., 2022) and NaCGEC (Ma et al., 2022), two widely-used benchmarks on CGEC. Training data follows previous work (Li et al., 2025), combining Lang8(Zhao et al., 2018), HSK(Zhang, 2009). Dataset statistics are provided in Appendix C.

Evaluation Metrics We report Precision (P), Recall (R), and $F_{0.5}$ scores using ChERRANT.

Baselines We compare against traditional Seq2Edit and Seq2Seq methods including GECToR (Omelianchuk et al., 2020), BART (Lewis et al., 2019), SynGEC (Zhang et al., 2022b), and MrGEC (Liu et al., 2024). For LLM-based approaches, we include Instruction Tuning (Liu et al., 2025), Alirector (Yang and Quan, 2024), DeCoGLM (Li and Wang, 2024), and direct prompting with DeepSeek-R1 (Qu et al., 2025). We also compare with preference optimization and reinforcement learning methods, including EPO (Liang et al., 2025), and rule-based RL with GRPO (Li et al., 2025), which represents the current state-of-the-art RL approach for GEC. The rule-based RL baseline undergoes the exact same two-stage SFT initialization as our method.

We use the GEC fine-tuned Qwen2.5-7B (Section 3.6) as the backbone for both EARM and GRPO training. Further implementation details of EARM and GRPO are provided in Appendix A,B.

4.2 Main Results

To evaluate our model, we consider two inference settings: (1) *single*, where the model generates one

Method	FCGEC			NaCGEC		
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
Traditional Methods						
GECToR	46.11	34.35	43.16	–	–	–
BART	63.07	39.95	56.53	62.04	45.84	57.94
SynGEC	63.75	39.78	56.89	62.42	47.41	58.71
MrGEC	65.71	37.78	57.22	–	–	–
LLM-based Methods						
Instruction Tuning	65.65	36.49	56.60	62.50	40.72	56.46
Alirector	64.49	36.22	55.78	66.93	46.59	61.55
EPO	66.67	41.93	59.63	67.09	**49.97**	62.79
DeCoGLM	56.09	38.02	51.22	–	–	–
Deepseek-R1	18.78	36.06	20.77	19.31	37.30	21.37
RL-based Methods						
Rule-based RL (single)	60.68	46.95	57.33	61.97	47.88	58.52
Rule-based RL (best-of-16)	62.84	**48.94**	59.46	63.24	49.68	59.97
EARM + GRPO (single)	68.98	42.39	61.29	69.78	45.57	63.08
EARM + GRPO (best-of-16)	**74.83**	42.68	**65.04**	**72.43**	45.08	**64.59**

Table 2: Results on Chinese GEC benchmarks. Best-of-16 denotes selecting the highest-scoring candidate among 16 rollouts using EARM. **Bold** indicates the best overall result; underline indicates the best result excluding Best-of-16.

output per input with temperature=0.6, and (2) *best-of-16*, where we sample 16 candidates with temperature=1, and use EARM to select the highest-scoring one as the final output.

Table 2 presents our main results on Chinese GEC benchmarks. EARM + GRPO (single) achieves state-of-the-art results among single-output methods, reaching 61.29 $F_{0.5}$ on FCGEC and 63.08 $F_{0.5}$ on NaCGEC, surpassing the previous best method EPO by 1.66 and 0.29 points respectively, and outperforming rule-based RL (single) by 3.96 and 4.56 points. The most notable improvement is in precision: 68.98 on FCGEC and 69.78 on NaCGEC, substantially higher than EPO (66.67 and 67.09) and other baselines, indicating that EARM effectively distinguishes high-quality corrections from plausible but incorrect alternatives. With best-of-16 reranking, performance further improves to 65.04 and 64.59 $F_{0.5}$, still outperforming rule-based RL (best-of-16) by 5.58 and 4.62 points, achieving +5.41 over EPO on FCGEC and +1.80 on NaCGEC, demonstrating EARM's additional value as a candidate selector. Notably, DeepSeek-R1 with direct prompting achieves only 20.77 $F_{0.5}$, indicating that even powerful LLMs require task-specific training for GEC.

Configuration	$F_{0.5}$
$\mathcal{L}_{\text{rank}}$ only	57.32
+ $\mathcal{L}_{\text{margin}}$	57.67
+ $\mathcal{L}_{\text{weighted}}$	58.15
+ sft_{base} (full EARM)	**58.61**

Table 3: Ablation on loss components.

4.3 Ablation Studies

We conduct ablation experiments on the FCGEC dataset to validate the contribution of each component. Specifically, we use the SFT model from Section 3.6 to generate 16 candidate corrections for each input on the FCGEC validation set. Different reward model variants—trained with various loss combinations and weighting strategies—are then used to score and select the best candidate for submission. We report $F_{0.5}$ scores to demonstrate the effectiveness of each proposed component.

Effect of Loss Components Seen from Table 3, using only sentence-level ranking loss achieves 57.32 $F_{0.5}$. Adding the margin loss improves performance to 57.67 (+0.35), providing explicit constraints for separating chosen and rejected candidates. The weighted token-level loss contributes a further gain to 58.15 (+0.48), validating that edit-aware token weighting helps the backbone learn

representations that better capture fine-grained correction quality. Finally, initializing from an SFT model trained on GEC data (sft_{base}) yields the best result of 58.61 (+0.46), indicating that task-specific pretraining provides a better starting point for reward modeling. All configurations in Tables 5, 6 share the same SFT-base initialization for fair comparison.

Edit Weight (α, γ)	$\mathbf{F_{0.5}}$
Uniform (1, 1)	56.89
(4, 2)	57.90
(3, 3)	57.76
(6, 3)	**58.15**
(8, 4)	57.91
(10, 5)	58.08

Table 4: Ablation on edit weights w/o SFT-base.

Effect of Edit Weights. As shown in Table 4, uniform weighting ($\alpha = \gamma = 1$) achieves only 56.89—even lower than the sentence-level-only baseline (57.32). This counterintuitive result suggests that naively applying token-level supervision without proper weighting introduces noise rather than useful signal, as unchanged tokens dominate the loss and dilute the edit-relevant gradients. Performance improves substantially when we upweight edit tokens: configurations such as $(6, 3)$, $(8, 4)$, and $(10, 5)$ all achieve strong results around 57.9–58.1, demonstrating that a reasonable range of weight settings can effectively guide the model to focus on edit-relevant regions. The comparison between $(3, 3)$ and $(6, 3)$ further confirms the importance of pivot weights—assigning greater importance to edit boundaries yields a notable improvement. Overall, these results validate the effectiveness of our category-based token weighting strategy: emphasizing edit-relevant tokens consistently improves reward modeling performance across a range of reasonable weight configurations.

Margin m	$\mathbf{F_{0.5}}$	**Coeff.** λ	$\mathbf{F_{0.5}}$
0.01	57.81	0.1	58.42
0.1	**58.61**	0.3	**58.61**
1.0	58.53	0.5	57.77

Table 5: Ablation on margin value m and weighted loss coefficient λ. Other hyperparameters are fixed at optimal values.

Effect of Margin Value m We search over margin values $m \in \{0.01, 0.1, 1.0\}$ (Left part in Table 5). A moderate margin ($m = 0.1$) achieves the best performance of 58.61. A too small margin ($m = 0.01$, 57.81) provides insufficient separation between chosen and rejected candidates, while a large margin ($m = 1.0$, 58.53) performs comparably but slightly worse. The optimal margin reflects the fine-grained nature of GEC evaluation, where candidate quality differences are often subtle.

Effect of Weighted Loss Coefficient λ We vary the coefficient λ for the weighted token-level loss in $\{0.1, 0.3, 0.5\}$ (Right part in Table 5). The optimal value $\lambda = 0.3$ achieves 58.61, balancing sentence-level and token-level supervision. A smaller coefficient ($\lambda = 0.1$, 58.42) slightly underweights the edit-aware signal, while a larger coefficient ($\lambda = 0.5$, 57.77) causes the token-level objective to dominate, potentially sacrificing global coherence for local edit accuracy.

5 Analysis

5.1 Reward Model Quality

Reward Type	**Accuracy**	**Kendall's** τ
PPL	59.54%	0.0522
Deepseek-R1	57.09%	0.1182
Sent-RM	75.69%	0.4605
EARM (full)	**81.33%**	**0.6608**

Table 6: Ranking accuracy on held-out preference pairs from FCGEC validation set. PPL: perplexity-based scoring using Qwen2.5-7B (lower perplexity = higher score). DeepSeek-R1: LLM-as-judge with pairwise comparison prompting. Sent-RM: reward model without edit-aware token weighting. Further implementation details of DeepSeek-R1 and Sent-RM are provided in Appendix A.

To directly evaluate reward model quality, we measure ranking accuracy on held-out preference pairs constructed from FCGEC validation set, as described in Section 3.4.

Table 6 shows that EARM achieves 81.33% accuracy and 0.6608 Kendall's τ, substantially outperforming all baselines. Perplexity-based scoring performs near random (59.54%), indicating that fluency alone is insufficient for GEC quality assessment. Surprisingly, DeepSeek-R1 as an LLM judge achieves only 57.09% accuracy, suggesting that even powerful LLMs struggle with fine-grained GEC discrimination without task-specific training. The sentence-level RM reaches 75.69%, demonstrating the value of learned rewards, but EARM's edit-aware weighting provides an additional 5.64%

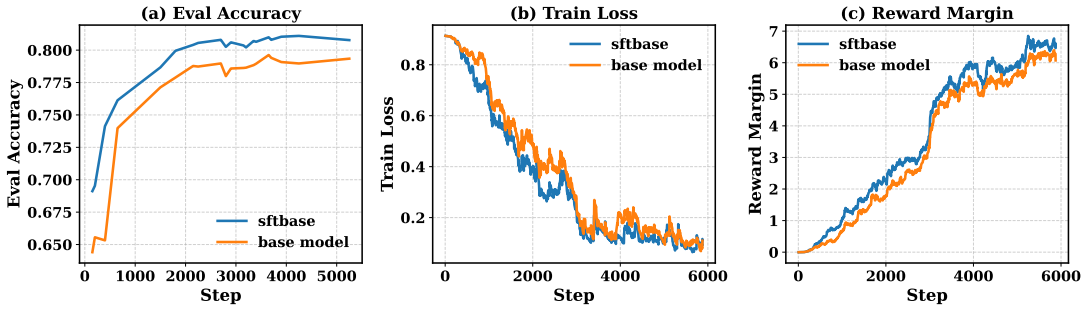

Figure 2: Training dynamics of EARM with SFT-base (GEC fine-tuned) and Base model (vanilla LLM) initializations. SFT-base achieves higher accuracy with stable convergence throughout training.

Edit Dist.	DS-R1	Sent-RM	EARM
≤ 1	55.55	76.10	**82.04**
$2-3$	61.08	74.98	**79.18**
≥ 4	69.49	66.10	**81.36**
Overall	57.09	75.69	**81.33**

Table 7: Ranking accuracy (%) stratified by edit distance difference between candidates. DS-R1: DeepSeek-R1 as judge. Sent-RM: sentence-level reward model without edit-aware weighting.

Method	$F_{0.5}$
SFT	53.94
DPO (same 188K pairs)	54.80
Rule-based RL	55.41
EARM + GRPO (Ours)	**58.61**

Table 8: Comparison with DPO on FCGEC validation set. DPO is trained on the same 188K preference pairs used for EARM. All methods share the same two-stage SFT initialization.

Error Type	P	R	$F_{0.5}$
Missing (M)	54.66	25.15	44.27
Redundant (R)	83.74	63.79	78.81
Substitution (S)	68.09	27.99	52.92
Word Order (W)	82.58	44.41	70.47

Table 9: EARM + GRPO performance by error type on FCGEC. M: insertion needed; R: deletion needed; S: replacement needed; W: reordering needed.

absolute improvement by explicitly focusing on correction-relevant tokens.

5.2 Fine-Grained Discrimination

To understand where EARM's advantage originates, we evaluate ranking accuracy on held-out preference pairs stratified by edit distance difference (Table 7). We compare three approaches: DeepSeek-R1 as an LLM judge, a sentence-level reward model trained without edit-aware token weighting, and our full EARM.

EARM consistently outperforms baselines across all difficulty levels, with the largest advantage on subtle distinctions ($\leq$ 1): 82.04% vs 76.10% for Sent-RM and 55.55% for DS-R1. Interestingly, Sent-RM's accuracy drops sharply for large edit differences ($\geq$ 4), while EARM maintains stable performance. This suggests that edit-aware weighting helps the model focus on correction-relevant signals regardless of overall edit magnitude.

5.3 Training Dynamics

Figure 2 visualizes EARM training dynamics under two backbone initializations: SFT-base (GEC fine-tuned Qwen2.5-7B) and Base model (vanilla Qwen2.5-7B). Both models exhibit stable convergence with smoothly decreasing loss and increasing reward margin, indicating improved discrimination over time. SFT-base consistently outperforms Base model and achieves higher final accuracy, suggesting that task-specific pretraining provides a better initialization for reward modeling. This motivates our final configuration using SFT-base as the backbone.

5.4 Comparison with DPO

To isolate the contribution of the reward signal from the optimization algorithm, we train a standard DPO baseline using the same 188K preference pairs as EARM on the FCGEC validation set. As shown in Table 8, DPO and rule-based RL achieve comparable performance (54.80 vs. 55.41), both modestly outperforming SFT, suggesting that simply applying preference optimization—whether offline (DPO) or online with binary rewards—yields limited gains beyond supervised learning. In contrast, EARM + GRPO substantially outperforms

Candidate	Edit Dist.	Rule	EARM
Source: 去年5月阿里巴巴宣布将旗下的一达通平台向我国外贸出口企业发放出口补贴……			
Reference: 去年5月阿里巴巴宣布将通过旗下的一达通平台向我国外贸出口企业发放出口补贴……			
……宣布将通过旗下的一达通平台向……(correct)	0	**1.0**	**3.3594**
……宣布将借助旗下的一达通平台向……(synonym)	1	0.0	3.2188
……宣布将旗下的一达通平台向……(unchanged)	1	0.0	2.0625
……宣布将通过旗下一达通平台向……(overcorrection)	1	0.0	0.6641
……宣布旗下的一达通平台向……(wrong edit)	2	0.0	-0.7656

Table 10: Case study comparing rule-based and EARM rewards. The source is missing a preposition before "旗下".

both DPO (+3.81) and rule-based RL (+3.20), demonstrating that the improvement is attributable to the quality of the edit-aware reward signal rather than the choice of optimization algorithm.

5.5 Error Type Analysis

Error Type Breakdown Table 9 shows the model performance on different error types. The model excels at redundant (78.81) and word order errors (70.47), with high precision on both. Missing word errors are most challenging (44.27), as the model is conservative about insertions.

Overcorrection Mitigation A key challenge for LLMs in GEC is overcorrection—modifying correct text unnecessarily. The high precision across all error types (54–84%) indicates that EARM effectively teaches the model when *not* to correct, avoiding the overcorrection tendency common in LLM-based GEC systems.

5.6 Case Study

Table 10 illustrates EARM's discrimination capability. Rule-based rewards assign 1 only to exact matches and 0 to all others, failing to distinguish synonyms, unchanged inputs, overcorrections, and wrong edits. EARM provides graded scores: acceptable synonyms receive high scores, unchanged inputs receive moderate scores, while overcorrections and wrong edits receive lower scores proportional to their severity.

6 Conclusion

We presented EARM, an edit-aware reward model that incorporates editing structure into preference learning through differentiated token weights and dual-granularity scoring. Integrated with GRPO, EARM achieves state-of-the-art performance on FCGEC and NaCGEC, substantially outperforming rule-based RL.

The key insight—that edit boundaries carry more discriminative information than unchanged tokens—may generalize to other text editing tasks such as text simplification and machine translation post-editing.

Limitations

Our work has the following limitations:

Alignment Tool Dependency: EARM relies on character-level alignment tools (ChERRANT) to identify edit spans. Although ChERRANT incorporates semantic similarity, phonetic similarity, and span-level merging to reduce alignment noise, alignment errors may still propagate to reward computation. We note that the edit mask only assigns token-level weights during RM training rather than serving as hard labels, so misclassified tokens cause slight weight perturbation but do not alter the preference ordering. Over 188K preference pairs, sporadic alignment errors are effectively averaged out during training.

Hyperparameter Sensitivity: The optimal edit weight configuration (α, γ) may vary across languages and error types. We leave automatic weight learning for future work.

Computational Cost: Training EARM requires constructing large-scale preference pairs (180K for Chinese), which involves multiple inference passes for candidate generation.

Language Coverage: Our experiments focus on Chinese. The effectiveness on other languages with different grammatical structures remains to be validated.

Acknowledgements

This work is supported by Beijing Natural Science Foundation (L253001), Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology) and National Engineering Research Center of New Electronic Publishing Technologies. We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs. *Biometrika*.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 643–701.

Steven Coyne, Keisuke Sakaguchi, Diana Galvan-Sosa, Michael Zock, and Kentaro Inui. 2023. Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction. *arXiv preprint arXiv:2303.14342*.

Tao Fang, Shu Yang, Kaixin Lan, Derek F. Wong, Jinpeng Hu, Lidia S. Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *Preprint*, arXiv:2304.01746.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. 2023. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Wei Li and Houfeng Wang. 2024. Detection-correction structure via general language model for grammatical error correction. *Preprint*, arXiv:2405.17804.

Yilin Li, Xunjian Yin, Yilin Chen, and Xiaojun Wan. 2025. Harnessing rule-based reinforcement learning for enhanced grammatical error correction. *Preprint*, arXiv:2508.18780.

Jiehao Liang, Haihui Yang, Shiping Gao, and Xiaojun Quan. 2025. Edit-wise preference optimization for grammatical error correction. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3401–3414, Abu Dhabi, UAE. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Xinpeng Liu, Bing Xu, Muyun Yang, Hailong Cao, Conghui Zhu, Tiejun Zhao, and Wenpeng Lu. 2025. A chain-of-task framework for instruction tuning of LLMs based on Chinese grammatical error correction. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8623–8639, Abu Dhabi, UAE. Association for Computational Linguistics.

Yumeng Liu, Zhenghua Li, HaoChen Jiang, Bo Zhang, Chen Li, and Ji Zhang. 2024. Towards better utilization of multi-reference training data for Chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3044–3052, Bangkok, Thailand. Association for Computational Linguistics.

Shirong Ma, Yinghui Li, Rongyi Sun, Qingyu Zhou, Shulin Huang, Ding Zhang, Li Yangning, Ruiyang Liu, Zhongli Li, Yunbo Cao, and 1 others. 2022. Linguistic rules-based corpus generation for native chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Andrii Liubonko, Oleksandr Skurzhanskyi, Artem Chernodub, Oleksandr Korniienko, and Igor Samokhin. 2024. Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models. *Preprint*, arXiv:2404.14914.

Fanyi Qu, Chenming Tang, and Yunfang Wu. 2025. Evaluating the capability of large-scale language models on chinese grammatical error correction task. *Preprint*, arXiv:2307.03972.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Lvxiaowei Xu, Jianwang Wu, Jiawei Peng, Jiayu Fu, and Ming Cai. 2022. FCGEC: Fine-grained corpus for Chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1900–1918, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Haihui Yang and Xiaojun Quan. 2024. Alirector: Alignment-enhanced Chinese grammatical error corrector. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2531–2546, Bangkok, Thailand. Association for Computational Linguistics.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Baolin Zhang. 2009. Features and functions of the hsk dynamic composition corpus (in chinese). *International Chinese Language Education*, (4).

Yue Zhang, Leyang Cui, Deng Cai, Xinting Huang, Tao Fang, and Wei Bi. 2023. Multi-task instruction tuning of llama for specific scenarios: A preliminary study on writing assistance. *Preprint*, arXiv:2305.13225.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022a. MuCGEC: a multi-reference multi-source evaluation dataset for Chinese grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130, Seattle, United States. Association for Computational Linguistics.

Yue Zhang, Bo Zhang, Zhenghua Li, Zuyi Bao, Chen Li, and Min Zhang. 2022b. SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2518–2531, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: Grammatical error correction. In *Natural Language Processing and Chinese Computing*, pages 439–445, Cham. Springer International Publishing.

Han Zhong, Guhao Feng, Wei Xiong, Xinle Cheng, Li Zhao, Di He, Jiang Bian, and Liwei Wang. 2024. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*.

We used Claude to assist with writing and editing this paper.

A Implementation Details

Reward Model Training We use Qwen2.5-7B as the backbone and perform full-parameter fine-tuning. The learning rate is set to 1×10^{-5} with linear warmup (10% of total steps) and cosine decay. We train with batch size 8 per GPU across 8 GPUs, with gradient accumulation steps of 8, yielding an effective batch size of 512. Training data consists of 188K preference pairs for Chinese.

The loss weights are $\lambda = 0.3$ for weighted token-level loss and $\mu = 0.1$ for margin loss, with margin value $m = 0.1$. Edit weights are set to $(\alpha, \gamma) = (6, 3)$ for pivot and continuation tokens respectively.

For comparison, we also train a sentence-level-only baseline by setting $\lambda = 0$, $\mu = 0$, and $m = 0$.

Reward Model Architecture The value head consists of two linear layers with dimensions $H \to H/2 \to 1$, where H is the hidden size of the backbone (4096 for Qwen2.5-7B). ReLU activation and 0.1 dropout are applied between layers. The token score head has dimensions $H \to H/4 \to 1$.

LLM-as-judge We use DeepSeek-R1 as an LLM judge to evaluate correction quality through pairwise comparison. Given a source sentence and two candidate corrections, the model is prompted to select the better one based on grammatical correctness and fluency. Figure 3 shows the prompt template used for this evaluation. We use this LLM-as-judge approach as a baseline to compare against EARM for evaluating reward model ranking quality.

Template for LLM-as-judge evaluation.

"role": "user", "content": '你是一个中文语法纠错专家。请判断以下两个纠错结果哪个更好。
原句：source
纠错结果A：response-a
纠错结果B：response-b
评判标准： 1. 语法正确性：纠正了原句的语法错误
2. 最小修改原则：只修改必要的错误，不做多余改动
3. 语义保持：保持原句的意思不变
请用 <ans> 标签给出你的判断，例如 <ans>A</ans> 或 <ans>B</ans>。如果两者质量相当，回答 <ans>TIE</ans>。
你的判断：'

Figure 3: Prompt template for LLM-as-judge evaluation.

Template for CGEC Reasoning

"role": "user", "content": '请识别我提供的句子是否有语法错误，如果有语法错误，请进行改正，请做出最少的修改。修改要求很严格，不要将流畅性，礼貌性，结构性，口语化，长短句，拗口性，风格等不属于语法范畴，而属于可优化的问题进行修改。如果没有错误，请回复原句。请保证你所做的修改都是有语法依据的，不要润色句子。 最终答案请你按照如下格式回复。
<think>
你的思考
</think>
<answer>
你修改后的句子，或者原句
</answer>
你要修改的句子如下：
[sentence]'

Figure 4: Template for CGEC reasoning SFT and RL

Template for evaluating data

你是一名语言学者，负责对语法纠错任务的解决程度评分。
我将向你提供一个需要修改的病句和这个病句的标准答案，你的工作是检查他人修改这个病句的时候的思考过程是否可以正确得出我所提供的答案，如果他的思考中缺少对我正确答案所提及的错误，或者他的思考认为标准答案认为正确的地方出现了错误，或者他的思考出现歧义，都视为不合理的思考。
如果你认为思考能得到我提供的答案请回复是，否则回复否。
注意，你需要模拟根据他思考过程得到的句子，之后与我的答案进行对比，如果不一样请回复否。
请你不要管我提供的标准答案是否还可以进一步优化，也不要管他的思考过程是多么的对，你只需要管这个思考过程能不能得到我提供的标准答案。
我将按照以下格式提供思考过程和正确答案：
需要修改的病句
ori
思考过程
think
标准答案
ans
解释你的推理，并在**新的一行**结束你的回答，此行只写“是”或“否”（不带引号）。
回复样例：
你的思考
是或否

Figure 5: Template for evaluating data

B GRPO Training Details

We use the verl[2] framework for GRPO training. The Clip-Higher strategy (Yu et al., 2025) is applied with asymmetric clipping bounds. We set KL coefficient to 0 as the SFT model already provides a good initialization. Prior to GRPO, we perform two-stage SFT warm-up following Li et al. (2025): Stage 1 uses Lang8 and HSK (5× replicated) augmented with reasoning traces from Qwen3-32B (1.57M pairs); Stage 2 continues on FCGEC with DeepSeek-R1 generated traces filtered by DeepSeek-V3 (27.5K instances). See Table 11, 12 for SFT and GRPO hyperparameters. The prompts for trace generation and filtering are shown in Figure 6 and Figure 5. The GEC instruction template is provided in Figure 4.

[2] https://github.com/volcengine/verl

Template for generating data

你的任务是展示对中文句子进行语法纠错的思考过程，但不展示已知的正确答案，也不展示你知道正确答案的事实。请仔细阅读以下信息，并按照指示输出思考过程。
待纠错的句子（可能没有错误）：
<待纠错句子>
src
</待纠错句子>
已纠正的句子（如果是正确的，就无需纠正）：
<已纠正句子>
tgt
</已纠正句子>
在展示思考过程时，请遵循以下要求：
1. 从语法规则的角度出发，分析原句可能存在的问题，如词性、语序、搭配、成分等。中文语法错误可归类为以下几类：语序不当、搭配不当、成分缺失、成分赘余、结构混乱、不合逻辑、语意不明 。
2. 阐述判断原句存在问题的依据。
3. 避免提及已知的正确答案内容。
4. 确保思考过程丰富、全面。
5. 如果没有语法错误，请不要随意润色句子。
请在<思考>标签内写下你的思考过程。
<思考>
在此详细展示对句子进行语法纠错的思考过程
</思考>
请在<答案>标签内写下你的答案。
<答案>
已纠正的句子
</答案>

Figure 6: Template for generating reasoning data

C Dataset Statistics

We evaluate on FCGEC (Xu et al., 2022) and NaCGEC (Ma et al., 2022), two widely-used benchmarks on CGEC. Training data follows previous work (Zhang et al., 2022b; Yang and Quan, 2024; Li et al., 2025), combining Lang8(Zhao et al., 2018), HSK(Zhang, 2009). Detailed statistics and usage are shown in Table 13.

D Hyperparameter Search

We perform grid search over the hyperparameters listed in Table 14 using FCGEC validation set. The optimal configuration is $\alpha = 6$, $\gamma = 3$, $\lambda = 0.3$, $\mu = 0.1$, $m = 0.1$.

Parameter	Stage 1	Stage 2
Base model	Qwen2.5-7B-Instruct	Stage 1 ckpt
Learning rate	1×10^{-5}	1×10^{-5}
Batch size	128	64
Warmup steps	3737	86
Epochs	3	2
Max length	2500	2500

Table 11: SFT training configuration.

Parameter	Value
Base model	Qwen2.5-7B (SFT)
Learning rate	1×10^{-6}
Batch size	128
Group size G	16
PPO clip ε_{low}	0.2
PPO clip ε_{high}	0.28
KL coefficient β	0
Temperature	1.0
Max generation length	2048
Training epochs	5
Hardware	8×A100 80GB
Training time	∼50 hours

Table 12: GRPO training configuration.

Dataset	#Sent	%Error	Usage
Chinese			
Lang8	1,220,906	89.5	Train
HSK	156,870	60.8	Train
FCGEC-train	36,341	54.3	Train
FCGEC-dev	2,000	55.1	Valid
FCGEC-test	3,000	–	Test
NaCGEC-test	5,869	95.6	Test

Table 13: Dataset statistics. #Sent denotes number of sentences, %Error denotes percentage of erroneous sentences.

Hyperparameter	Search Range
Edit weight α (pivot)	{4, 6, 8, 10}
Edit weight γ (continuation)	$\alpha/2$
Weighted token-level loss λ	{0.1, 0.3, 0.5}
Margin loss μ	0.1 (fixed)
Margin value m	{0.01, 0.1, 1.0}
Learning rate	1×10^{-5} (fixed)

Table 14: Hyperparameter search space for EARM training.