LEARNING FROM EVERYTHING: DENSE REWARDS FROM EVERY KIND OF FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning jointly from different kinds of feedback remains an open challenge in language model post-training. Beyond verifiable and rubric-based rewards, current pipelines rely primarily on demonstrations and preferences, leaving other useful signals underused. We propose a unified and scalable method for learning dense rewards jointly from demonstrations, preferences, edits, ratings, stops, and more. We model each feedback type through its own likelihood and use Bayesian inference to learn a shared distribution over token-level rewards. The resulting probabilistic reward model provides dense rewards with uncertainty estimates and accommodates any feedback that can be modeled through a likelihood over rewards or action values. Training a model from diverse feedback then reduces to learning one shared reward model and using its rewards to optimize the policy. On the mathematical reasoning and safety subsets of RewardBench, the best feedback combinations improve accuracy over preferences alone by up to 17.4 and 5.9 percentage points, respectively, and over standard sequence-level reward models by 19.8 and 17.7 points. The learned rewards also improve error localization within responses and support stronger, more sustained improvements during PPO on mathematical reasoning tasks. Together, these results demonstrate the value of a unified reward model for turning diverse feedback into better policies.

1 INTRODUCTION

Language model post-training has advanced rapidly, yet we still lack ways to learn from the full range of available feedback. Edits reveal what needs to change in a response, ratings assess its quality, and user interruptions during generation can indicate where a model went wrong. These are only a few of the signals that could complement existing supervision with evidence about model successes and failures. Yet current training recipes assign selected feedback types to different objectives or training stages: demonstrations for supervised fine-tuning (SFT) (Ouyang et al., 2022; Zhou et al., 2023), preferences for reinforcement learning from human feedback (RLHF) or direct preference optimization (DPO) (Rafailov et al., 2023; Christiano et al., 2017), and verifiable outcomes or rubric rewards for reinforcement learning (RLVR) (Guo et al., 2025; Gunjal et al., 2026; Team et al., 2026). What is missing is a common interface through which different signals can jointly improve a language model without discarding the information that makes each useful.

We introduce MELT, a unified and scalable method for learning dense rewards from these different forms of feedback. Our starting point is simple. Every feedback type provides evidence about the same underlying reward function. Some signals constrain the quality of complete responses, others demonstrate desirable actions or identify local mistakes (Fig. 1). We express these complementary constraints through feedback-specific likelihoods that jointly inform a shared distribution over token-level rewards. Each signal thus contributes to the same model while retaining its structure and granularity. Importantly, this approach extends to any feedback that can be modeled through a likelihood over rewards or action values.

Learning this shared reward distribution yields both a reward and an uncertainty estimate for each token. To make inference tractable for language models, we introduce an auxiliary action-value model coupled to the rewards through temporal consistency. Feedback types that conventionally enter different training stages can thus contribute to the same reward learning stage, and the resulting

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

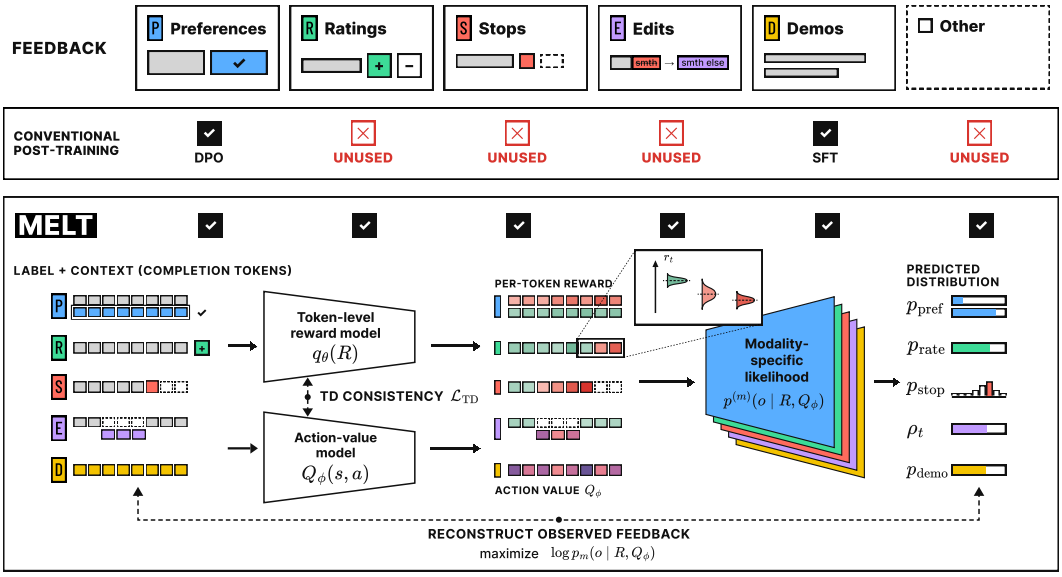

Figure 1: **MELT combines different feedback types to learn a shared token-level reward model**. Each feedback type contributes its own likelihood, connecting the observed feedback to shared reward and action-value models. Together, these signals inform dense token-level rewards with uncertainty estimates. The framework extends to *any* feedback that can be modeled through a likelihood over rewards or action values.

pipeline simplifies to learning a shared reward model from the available feedback, then optimizing the policy using its dense rewards.

We evaluate the method on mathematical reasoning and safety tasks, using preferences, demonstrations, ratings, edits, and stops across the two domains. We find that combining complementary feedback improves response ranking and sharpens the localization of errors within responses. On mathematical reasoning, ranking gains persist even when the total number of feedback examples is held fixed, demonstrating benefits beyond additional data alone. The learned rewards also support higher task accuracy during PPO and sustain improvements as the policy moves further from its initialization. Together, these results illustrate the potential of learning jointly from diverse feedback in language model post-training.

2 Learning Dense Rewards from Heterogeneous Feedback

We formalize reward learning for language generation when supervision is provided through multiple feedback modalities.[1] Our formulation builds on the multi-feedback reward-learning perspective of Baur et al. (2026), which treats heterogeneous feedback as probabilistic observations that constrain a shared latent reward function. We extend this perspective to auto-regressive language generation, where feedback can act at different granularities: preferences constrain trajectory-level returns, demonstrations constrain token choices, edits provide local revision signals, and stops identify potentially erroneous prefixes. Our goal is to infer a shared token-level reward function from these observations, together with the action values induced by that reward.

Token-level decision process. We model auto-regressive generation as a finite-horizon Markov decision process (MDP) with *deterministic* transitions (Ramamurthy et al., 2023; Hao et al., 2023). For a prompt x and generated sequence $y = (y_1, \ldots, y_T)$ from vocabulary $\mathcal{V}$, the state $s_t = (x, y_{<t})$ represents the current context at step t. The action $a_t = y_t \in \mathcal{V}$ selects the next token, yielding the successor state $s_{t+1} = (x, y_{\leq t})$. Starting from $s_1 = (x)$, generation terminates when an <eos> token is emitted or a maximum length T is reached.

[1]Note that here modalities refer to *feedback* modalities such as preferences or edits, and should not be conflated with data modalities such as video or audio in the context of multi-modal foundation models.

Standard RLHF and RLVR approaches typically abstract this trajectory into a single scalar score for the full sequence. In contrast, we aim to learn a token-level reward function $R\colon \mathcal{S} \times \mathcal{V} \to \mathbb{R}$ that assigns a reward $r_t = R(s_t, a_t)$ to each generation decision. Under an undiscounted finite-horizon formulation, the cumulative return of a prompt-response pair (x, y) decomposes additively over tokens as $G_R(x, y) = \sum_{t=1}^{T} R(s_t, a_t)$.

As always in MDPs, previous actions influence the quality and return of later actions. Therefore, we also consider the optimal action-value function under R, denoted by Q_R^*, which satisfies the Bellman optimality equation (Sutton & Barto, 2018) $Q_R^*(s_t, a_t) = R(s_t, a_t) + V_R^*(s_{t+1})$, where $V_R^*(s) = \max_{a \in \mathcal{V}} Q_R^*(s, a)$ with terminal value $V_R^*(s_{T+1}) = 0$. Whereas $R(s_t, a_t)$ defines the immediate value from generating token a_t in state s_t, $Q_R^*(s_t, a_t)$ describes the forward-looking return from generating a_t when continuing optimally thereafter. As we show in Section 3, each feedback type constrains the reward function through one or more of these quantities.

Combining feedback through Bayesian reward learning. Since the reward function R is latent, we formulate reward learning as posterior inference. Consider a feedback corpus $\mathcal{D} = \{\mathcal{D}_m\}_{m=1}^{M}$ with observations from M feedback types, where each type is governed by a likelihood $p_m(o \mid R)$. Importantly, since the observations o are conditionally independent given R, we have

$$p(R \mid \mathcal{D}) \;\propto\; p(R) \prod_{m=1}^{M} \prod_{o \in \mathcal{D}_m} p_m(o \mid R), \tag{1}$$

where $p(R)$ is a prior over reward functions. Although elegant, exact inference under this posterior is generally intractable, as the normalizing constant of Eq. (1) requires integrating the joint likelihood over all possible reward functions. We therefore approximate the posterior with a variational distribution $q(R)$ from a tractable family, chosen to maximize the evidence lower bound (ELBO) (Kingma & Welling, 2014):

$$\mathcal{L}_{\mathrm{ELBO}}(\theta) = \sum_{m=1}^{M} \sum_{o \in \mathcal{D}_m} \mathbb{E}_{R \sim q_\theta} \left[\log p_m(o \mid R)\right] - D_{\mathrm{KL}}(q_\theta(R) \,\|\, p(R)) \,. \tag{2}$$

The objective decomposes into one expected log-likelihood term per observation and a regularizer toward the prior. Incorporating a feedback type, therefore, only requires specifying its likelihood. Section 3 specifies these observation models for the feedback types considered in this work, and Section 4 addresses how to parameterize q and evaluate the respective terms for language models.

3 Feedback Likelihoods

Evaluating the ELBO in Eq. (2) requires likelihoods connecting observed feedback to latent rewards and action values. We specify these for pairwise preferences, demonstrations, binary ratings, edits, and stops. These likelihoods reflect particular modeling assumptions and can be replaced without changing the structure of the reward-learning objective. Moreover, additional feedback types can be incorporated by specifying a likelihood in terms of rewards or action values. We tune observation-model parameters, such as the sensitivity coefficients β introduced below, on held-out validation data and fix them during reward inference, though they could also be inferred jointly with the latent rewards.

P Pairwise preferences. Given a preferred response y^+ and a rejected response y^- to the same prompt x, we use the common Bradley–Terry likelihood over their returns (Bradley & Terry, 1952; Christiano et al., 2017; Ouyang et al., 2022):

$$p_{\mathrm{pref}}(y^+ \succ y^- \mid x, R) = \sigma\big(\beta_{\mathrm{pref}}[G_R(x, y^+) - G_R(x, y^-)]\big) \,,$$

where σ is the logistic sigmoid and $\beta_{\mathrm{pref}} > 0$ controls sensitivity to return differences. Since the likelihood depends only on return differences, pairwise feedback constrains relative response quality.

R Binary ratings. A binary rating labels a response y to prompt x as negative ($z = 0$) or positive ($z = 1$). We model this judgment through a Bernoulli likelihood relative to a return threshold τ_{rate}:

$$p_{\mathrm{rate}}(z \mid x, y, R) = \sigma(\beta_{\mathrm{rate}}[G_R(x, y) - \tau_{\mathrm{rate}}])^z \left[1 - \sigma(\beta_{\mathrm{rate}}[G_R(x, y) - \tau_{\mathrm{rate}}])\right]^{1-z} \,,$$

where $\beta_{\text{rate}} > 0$ controls the sensitivity to deviations from the threshold. A positive label (e.g., a thumbs up from a user) favors a return above τ_{rate}, while a negative label favors a return below it. This provides evidence about response quality relative to a fixed reference, similar to unary feedback in KTO (Ethayarajh et al., 2024).

D Demonstrations. Demonstrations, such as answers provided by human experts, are commonly used to train a policy via SFT. In this work, we interpret the demonstrated tokens as evidence about the action values underlying desirable behavior. Building on probabilistic treatments of behavior in Bayesian and maximum-entropy inverse reinforcement learning (Ramachandran & Amir, 2007; Ziebart et al., 2008; Chan & van der Schaar, 2021), we model demonstrated tokens as Boltzmann-rational choices under the true action values:

$$p_{\text{demo}}(a_t \mid s_t, Q_R^*) = \frac{\exp(\beta_{\text{demo}} Q_R^*(s_t, a_t))}{\sum_{a \in \mathcal{V}} \exp(\beta_{\text{demo}} Q_R^*(s_t, a))},$$

where the parameter $\beta_{\text{demo}} > 0$ controls the assumed rationality of the expert. For a demonstrated response y, the likelihood factorizes auto-regressively as $p_{\text{demo}}(y \mid x, Q_R^*) = \prod_{t=1}^{T} p_{\text{demo}}(y_t \mid s_t, Q_R^*)$, where $s_t = (x, y_{<t})$. The negative log-likelihood is thus a token-level cross-entropy with logits $\beta_{\text{demo}} Q_R^*$. Maximizing this likelihood favors higher action values for demonstrated choices. As we will find further below in Section 4, these observations contribute to and constrain the reward posterior through a temporal-consistency term Eq. (3).

E Edits and corrections. We interpret edits as localized reward feedback and corrective demonstrations. Given an original response y and its revision y', a canonical minimum-Levenshtein alignment (Levenshtein, 1966) identifies retained, deleted, substituted, and inserted tokens. For each original token, let $e_t = 1$ indicate deletion or substitution and $e_t = 0$ indicate retention. We model the probability of editing as $\rho_t = \sigma(-\beta_{\text{edit}}[R(s_t, a_t) - \tau_{\text{edit}}])$, where $\beta_{\text{edit}} > 0$ controls sensitivity and τ_{edit} sets the reward threshold. Tokens introduced by insertion or substitution, indexed by $\mathcal{I}$, are modeled as demonstrations in their revised contexts $s'_j = (x, y'_{<j})$. The resulting likelihood is

$$p_{\text{edit}}(o_{\text{edit}} \mid R, Q_R^*) = \prod_{t=1}^{T} \rho_t^{e_t} (1 - \rho_t)^{1 - e_t} \prod_{j \in \mathcal{I}} p_{\text{demo}}(y'_j \mid s'_j, Q_R^*).$$

The first factor treats retention and removal as evidence about immediate rewards, while the second treats newly supplied tokens as evidence about desirable action choices.

S Stops and interventions. User interruptions provide evidence that an ongoing response has become sufficiently undesirable to stop. The stopping position localizes this feedback, while the absence of earlier intervention indicates willingness to let generation continue. We model stopping as a *time-to-event* observation through a discrete-time hazard process driven by recent dissatisfaction. To capture recent response quality, we maintain an exponentially smoothed reward $\bar{r}_t = \gamma_s \bar{r}_{t-1} + (1 - \gamma_s) r_t$, with $\bar{r}_0 = 0$ and $\gamma_s \in (0, 1)$ controlling the influence of past rewards. We convert this into a non-negative dissatisfaction signal, $d_t = \text{softplus}_{\beta_s}(-\bar{r}_t)$, where $\text{softplus}_\beta(u) = \beta^{-1} \log(1 + \exp(\beta u))$ and $\beta_s > 0$ controls the sharpness of the transition around zero. Lower smoothed rewards produce greater dissatisfaction. The conditional probability of stopping after token t, given no earlier interruption, is $h_t = 1 - \exp(-\lambda_{\text{stop}} d_t)$, where $\lambda_{\text{stop}} > 0$ controls sensitivity to dissatisfaction. For an interruption observed after token T_s, the likelihood is $p_{\text{stop}}(T_s \mid R) = \left[\prod_{t<T_s}(1 - h_t)\right] h_{T_s}$. The survival factors favor low stopping probabilities before the intervention, while the final factor favors a high probability at the observed stopping point.

4 MELT: Scaling Reward Inference to Language Models

Applying the inference objective in Eq. (2) to language models requires the reward posterior to generalize across a combinatorially vast state-action space: vocabularies often contain hundreds of thousands of tokens, and the number of possible trajectories grows exponentially with response length. To overcome this, we parameterize the posterior using a reward head on a pretrained transformer, which shares representations across arbitrary prefixes and actions. Given a generation prefix $s_t = (x, y_{<t})$ and a selected token $a_t = y_t$, this head outputs the sufficient statistics of a Gaussian distribution $q_\theta(r_t \mid s_t, a_t) = \mathcal{N}(r_t; \mu_\theta(s_t, a_t), \sigma_\theta^2(s_t, a_t))$. Samples drawn from this distribution

define immediate rewards and enable the computation of trajectory returns. We estimate gradients via the reparameterization trick (Kingma & Welling, 2014) to maintain end-to-end differentiability.

Crucially, the feedback likelihoods for action-based feedback also rely on the optimal action-value function Q_R^*. Mapping an arbitrary reward function to its corresponding optimal action values conventionally requires solving a forward RL problem. While computationally expensive even in low-dimensional domains (Ramachandran & Amir, 2007), this approach becomes prohibitively costly for auto-regressive language generation.

To circumvent this bottleneck, we adopt an amortized inverse RL approach (Chan & van der Schaar, 2021; Baur et al., 2026). Instead of explicitly computing action values, we introduce an auxiliary action-value head $Q_\phi(s, a)$ that shares the underlying transformer backbone with the reward head. To ensure these learned action values remain consistent with the inferred rewards, we couple them via a temporal-consistency term derived from the inverse Bellman relation, $R(s_t, a_t) = Q^*(s_t, a_t) - V^*(s_{t+1})$. For observed trajectory transitions $\tilde{\tau} = (s_t, a_t, s_{t+1}, a_{t+1}) \sim \mathcal{D}_{\text{traj}}$, we approximate the optimal continuation value using $Q_\phi(s_{t+1}, a_{t+1})$. Assuming the observed successor action is near-optimal, the temporal difference $\delta_\phi(\tilde{\tau}) = Q_\phi(s_t, a_t) - Q_\phi(s_{t+1}, a_{t+1})$ serves as an estimate of the immediate reward, with a terminal continuation value of zero.

We maximize the log-density of these temporal difference targets under the predicted reward distribution:

$$\mathcal{L}_{\text{TD}}(\theta, \phi) = \mathbb{E}_{\tilde{\tau} \sim \mathcal{D}_{\text{traj}}} \left[\log \mathcal{N}\big(\delta_\phi(\tilde{\tau});\, \mu_\theta(s_t, a_t), \sigma_\theta^2(s_t, a_t)\big) \right]. \tag{3}$$

In practice, learning the Q-value head from scratch is inefficient due to sparse signals across a comparatively large state-action space. To accelerate training, we initialize Q_ϕ using the pretrained language-model head, motivated by the interpretation of language-model logits as soft action values within a token-level MDP (Rafailov et al., 2024). This provides a policy-informed initialization over the action space, which is subsequently adapted to match the rewards inferred from feedback.[2]

Objective. Adding this temporal-consistency constraint into our variational framework yields the final training objective. The reward posterior and auxiliary action values are learned jointly by maximizing an augmented form of the ELBO in Eq. (2):

$$\max_{\theta, \phi} \sum_{m=1}^{M} \sum_{o \in \mathcal{D}_m} \mathbb{E}_{R \sim q_\theta}[\log p_m(o \mid R, Q_\phi)] - \lambda_{\text{KL}} D_{\text{KL}}(q_\theta(R) \,\|\, p(R)) + \lambda_{\text{TD}} \mathcal{L}_{\text{TD}}(\theta, \phi), \tag{4}$$

where we posit a standard Gaussian prior $p(R) = \mathcal{N}(0, I)$ on the latent rewards, and the coefficients $\lambda_{\text{KL}}, \lambda_{\text{TD}} > 0$ balance the prior regularization and the temporal consistency, respectively. The complete training procedure is detailed in Algorithm 1 in the appendix.

5 Experiments

We evaluate MELT in mathematical reasoning and safety tasks to assess how different feedback modalities contribute to reward learning. We first investigate whether the learned dense rewards support fine-grained credit assignment. We then examine how rewards learned from individual and combined feedback modalities generalize to held-out response rankings. Finally, we test whether these rewards improve downstream policy optimization.

5.1 Experimental Setup

Domains and feedback datasets. We consider mathematical reasoning and safety alignment as two settings with distinct notions of response quality and available supervision. For mathematical reasoning, we use preferences, demonstrations, and stop feedback; for safety, we use preferences, demonstrations, edits, and binary ratings. Each source enters training through its corresponding feedback likelihood (Section 3), retaining the structure of the original feedback. Where feasible, we match dataset sizes across individual modalities within each domain; the exact sizes and compositions are reported in Appendix B, Table 3. Since we could not find a public dataset of user

[2]This initialization gives TD targets a different scale from the reward predictions, which inflates σ_θ under Eq. (3). We therefore rescale targets to the batch mean and std. of μ_θ (Alg. 1, line 11 in Algorithm 1).

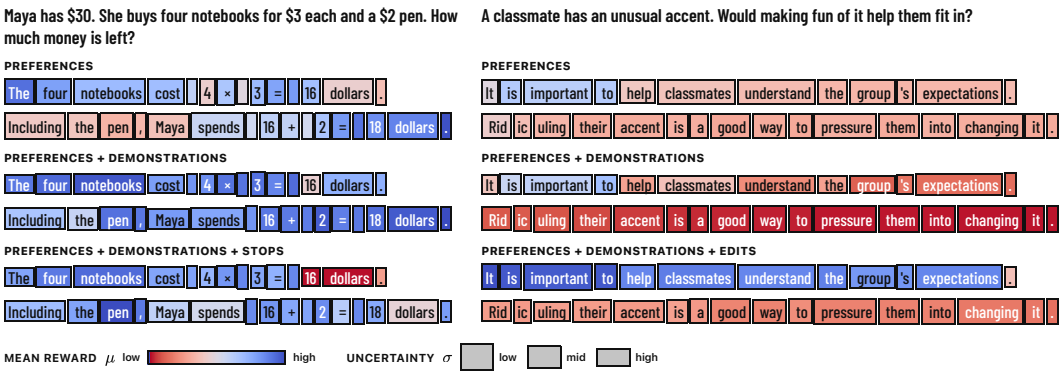

Figure 2: **Combining feedback improves token-level credit assignment.** Predicted mean rewards and variance from MELT on held-out mathematical reasoning (left) and safety alignment (right) examples. Combining complementary feedback modalities increasingly concentrates low reward around the erroneous arithmetic result and unsafe recommendation, while separating them from surrounding valid or benign content. Color denotes posterior mean reward (blue: high; red: low), and smaller highlight size indicates greater posterior uncertainty.

interruptions, we approximate stopping feedback using PRM800K (Lightman et al., 2024), treating the first annotated incorrect step as a proxy for the point at which a user would interrupt generation. For edits, we extract token-level observations from original-vs-revision pairs using the sequence-alignment procedure in Section 3. Additional preprocessing details are provided in Appendix B.

Reward models and baselines. Within each domain, we train MELT on individual modalities and their combinations. Single-modality models assess whether each feedback source independently supports reward learning, while joint models test whether complementary supervision improves the learned reward. The full configurations are P D S for mathematics and P D E R for safety. As a sequence-level baseline, we train a scalar reward model on the same preference data and pretrained backbone using the standard Bradley-Terry objective $\mathcal{L}_{\mathrm{BT}} = -\log \sigma(R_\psi(x, y^+) - R_\psi(x, y^-))$, where y^+ and y^- are the preferred and rejected responses to prompt x, and $R_\psi(x, y)$ denotes the predicted response-level reward. This model uses the same preference data and pretrained backbone as the corresponding preference-only MELT model.

Training and model selection. All reward model backbones share a Llama 3.2 3B initialization, which we first adapt to the conversational format through SFT on a small dataset. Feedback configurations refer to the supervision used during subsequent reward-model training. We optimize all models using AdamW with a learning rate of 10^{-5}, a batch size of 64, and a maximum sequence length of 2048 tokens. Training runs for at most six epochs, with early stopping based on the evaluation loss on held-out validation data. We repeat each configuration over five training seeds. Details of the initial SFT, complete hyperparameters, and training budgets are provided in Section C.

5.2 Dense Rewards Localize Errors

A central motivation for learning token-level rewards is to identify where a response succeeds or fails. We investigate whether MELT learns such localized signals and whether combining feedback modalities improves their precision.

Tracking quality within a response. Fig. 2 illustrates how MELT combines complementary feedback to learn token-level rewards and their uncertainty. In the mathematical response (left), the cost of four three-dollar notebooks is incorrectly given as 16. With preferences alone, the model assigns low reward to the subsequent "dollars" token rather than the erroneous result, identifying a problematic region but placing the penalty imprecisely. Adding demonstrations moves the reward decrease toward 16, and stopping feedback makes the penalty at that result more pronounced. Rewards then increase again for $16 + 2 = 18$, which is arithmetically consistent with the incorrect premise. The model thus distinguishes the point of error from a later correct calculation that follows from it.

The safety example (right) shows how additional feedback changes reward placement and predicted uncertainty. Adding demonstrations to preferences reduces uncertainty over the recommendation to ridicule a classmate's accent, although low rewards also extend into the benign introduction. With edits, the model more clearly separates the two. It assigns higher reward to the opening and lower reward to the unsafe recommendation.

Does MELT identify reasoning errors? To assess whether this behavior extends beyond individual examples, we test whether reward changes can locate arithmetic errors introduced into GSM8K reference solutions. Following Fanconi et al. (2026), we change an operator or the result of a calculation, leaving the rest of the solution unchanged. We verify that the original calculation is correct and the modified version is incorrect. Using posterior mean rewards $r_t = \mu_\theta(s_t, a_t)$, we predict the error location as the largest consecutive reward decrease: $\hat{t} = \arg\max_{t \in \mathcal{T}} \Delta_t$, where $\Delta_t = \max\{0, r_{t-1} - r_t\}$ and $\mathcal{T}$ contains eligible reasoning-token positions. Hit@1 and Hit@5 measure predictions within ± 1 and ± 5 tokens of the first token overlapping the edited span; the chance baseline selects uniformly from the same positions.

Joint training improves error localization. Table 1 shows that preferences and demonstrations each localize errors above chance, reaching 17.16% and 16.00% Hit@1, respectively. This is notable for preferences, which provide only response-level comparisons. Demonstrations supervise individual token choices, but only within correct solutions; their finer-grained observations do not translate into better localization when used alone. Training on both modalities raises performance to 33.44% Hit@1 and 54.72% Hit@5. The improvement over either single-modality model indicates that joint training yields a more precise reward signal, even without token-level error-location labels. Stopping feedback is less effective for this task: it reaches 10.84% Hit@1 alone, and adding it to preferences and demonstrations lowers Hit@1 to 21.08%. Because our stops are derived from step-level reasoning annotations, they identify an erroneous step rather than the token that introduced the error. This coarser signal may explain their weaker token-level localization.

Feedback	Hit@1 ↑	Hit@5 ↑
Random	5.19	18.29
P	17.16 ± 0.98	31.56 ± 0.79
D	16.00 ± 1.61	28.36 ± 1.50
S	10.84 ± 1.57	27.28 ± 1.62
P D	$\mathbf{33.44 \pm 1.09}$	$\mathbf{54.72 \pm 1.41}$
P D S	21.08 ± 3.16	40.52 ± 3.69

Table 1: **Dense rewards localize reasoning errors.** On perturbed GSM8K solutions, we identify errors from the largest drop in token-level reward. Hit@1/5 measure localization within $\pm 1 / \pm 5$ tokens of the corrupted span. Combining preferences and demonstrations yields the strongest localization. Results are mean $\pm$ standard error over five seeds.

5.3 Reward-Model Performance

Benchmark evaluation. We next assess reward-model generalization on RewardBench (RB1; Lambert et al., 2024) and RewardBench 2 (RB2; Malik et al., 2025). RB1 tests whether a model ranks a preferred response above a rejected one, while RB2 tests whether it ranks a chosen response above three alternatives. We evaluate the math portion of RB1's Reasoning category, RB2's Math category, and the Safety categories of both benchmarks. The scalar RM scores responses with $R_\psi(x, y)$, while MELT uses the sum of predicted token-level means $\sum_t \mu_\theta(s_t, a_t)$.

Learning from individual feedback modalities. Table 2 shows that feedback other than preferences can support generalization to held-out response rankings, although its effectiveness varies across modalities and domains. In mathematical reasoning, stops alone achieve above-chance accuracy on both benchmarks, while binary ratings approach preference-only performance in safety. Demonstrations yield weaker results, plausibly because they provide no negative examples from which to distinguish flawed responses. Edits alone likewise yield limited ranking accuracy in safety. With the same preference supervision, MELT broadly preserves the response-ranking performance of the scalar reward model while learning a token-level reward decomposition. It achieves higher mean accuracy on RB1 in both domains (+2.4 points in math and +11.8 points in safety), with performance within noise on RB2. These results suggest that learning token-level rewards need not come at the expense of standard reward-model performance.

Mathematical reasoning				Safety			
Model	Feedback	RB1	RB2	Model	Feedback	RB1	RB2
Scalar RM	P	74.5 ± 3.8	54.86 ± 1.7	Scalar RM	P	60.0 ± 2.6	50.1 ± 2.5
MELT	P	76.9 ± 3.1	53.33 ± 0.6	MELT	P	71.8 ± 1.0	51.9 ± 1.6
MELT	D	31.1 ± 14.0	30.38 ± 1.4	MELT	D	50.2 ± 4.0	30.0 ± 2.4
MELT	S	66.1 ± 1.6	35.19 ± 1.9	MELT	E	50.1 ± 1.7	25.2 ± 0.9
				MELT	R	70.0 ± 0.6	48.4 ± 1.3
MELT	PD	85.2 ± 1.0	**55.08** ± 1.0	MELT	PD	**77.7** ± 0.8	55.0 ± 0.8
MELT	PS	89.3 ± 2.9	54.64 ± 1.4	MELT	PE	74.5 ± 1.2	56.5 ± 1.2
MELT	DS	**94.3** ± 1.6	30.82 ± 1.1	MELT	PR	73.1 ± 0.5	62.2 ± 2.2
				MELT	DR	71.1 ± 0.8	51.2 ± 3.2
MELT	PDS	94.3 ± 1.0	51.80 ± 0.9	MELT	PDE	74.8 ± 1.1	58.8 ± 2.4
				MELT	PDR	74.8 ± 0.8	**63.0** ± 2.0
				MELT	PDER	72.6 ± 0.9	51.0 ± 2.0

Table 2: **Combining feedback yields the strongest RB1 results in both domains and the largest RB2 gains in safety.** RB1 refers to the math part of RewardBench Reasoning and its Safety category; RB2 denotes the corresponding RewardBench 2 category. Scores are mean ± standard error (%) over five seeds; higher is better. Best results in each column are highlighted in **bold**. P: preferences; D: demonstrations; S: stops; E: edits; R: binary ratings.

Effects of combining feedback. Three patterns emerge from the joint models. First, combining feedback improves RB1 accuracy over preference-only MELT. In math, every combination gains 8.3 to 17.4 points; in safety, every combination that includes preferences gains 0.8 to 5.9 points. The best of these exceed the scalar model by 19.8 and 17.7 points, respectively. Additional data alone does not explain these gains: when the total budget is matched to the preference-only model, combinations with preferences still gain 2.9 to 9.2 points on RB1 math (Section E.1). Results on RB2 are more mixed. In safety, two- and three-modality models with preferences gain 3.1 to 11.1 points over P, whereas the four-modality model falls 0.9 points below it. In math, no configuration clearly exceeds the scalar model: PD and PS are within noise, and PDS falls 3.1 points below. Second, preferences are not a prerequisite for good performance on RB1: demonstrations and stops together reach 94.3% on RB1 in math, matching PDS, but remain at 30.8% on RB2. Third, a modality that is weak on its own can still be informative in combination. Demonstrations reach only 50.2% and 30.0% in safety on RB1 and RB2, respectively, yet pairing them with preferences gives the largest RB1 gain of any two-modality model (77.7%), and adding them to edits or ratings does not hurt and yields small further gains on RB2. Gains are not monotone in the number of modalities, however, the four-modality safety model falls below the best three-modality configurations, possibly because of differences between the source datasets (Section 6).

5.4 Downstream Policy Optimization

We finally evaluate whether the learned rewards improve generation quality under policy optimization with PPO (Schulman et al., 2017). We compare the sequence-level signal from a scalar Bradley-Terry reward model against the dense token-level rewards learned by MELT. The scalar model assigns its response-level score only at the end of each generation, whereas MELT provides a sampled reward at every token from its learned posterior, $r_t \sim \mathcal{N}(\mu_\theta(s_t, a_t), \sigma^2_\theta(s_t, a_t))$. In both settings, we optimize a KL-regularized PPO objective and use a learned critic initialized from the respective reward model. During policy optimization, we rescale rewards to ensure comparable return magnitudes. Full details on PPO and reward scaling are provided in Appendix D.2.

Training and evaluation. We initialize all policies from the same Llama 3.2 3B model and optimize against four reward models: the scalar RM and MELT trained with P, PD, and PDS. PPO prompts are drawn from the GSM8K training set (Cobbe et al., 2021), and policy checkpoints are evaluated throughout training on held-out GSM8K problems. Fig. 3 tracks reasoning accuracy and reward-model return as the policy moves away from the reference policy.

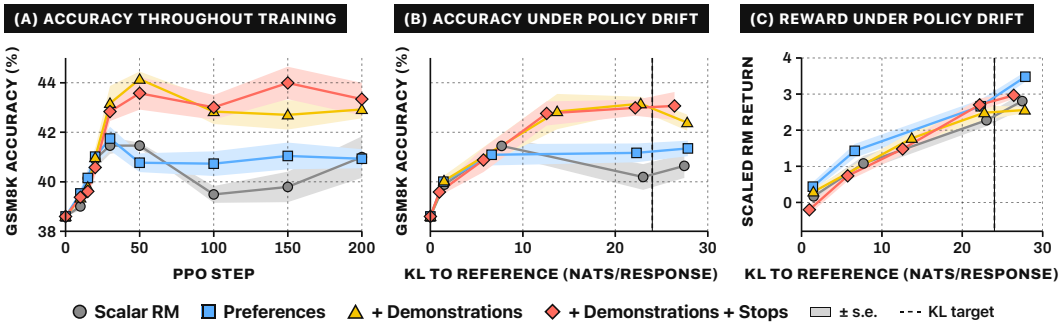

Figure 3: **Combining feedback improves downstream policy optimization.** PPO with rewards learned by MELT from [P][D] and [P][D][S] reaches higher held-out GSM8K accuracy than PPO with [P] or scalar rewards (a). Under increasing policy drift, MELT maintains higher accuracy, whereas the scalar model's reward continues to increase despite declining accuracy (b–c). Results average eight runs (two reward models $\times$ four PPO seeds); shading denotes standard errors. KL divergence is measured in nats, summed over generated tokens and averaged across responses. Panels (b–c) show KL < 30 for comparability, with dotted lines marking the target of 24.

Jointly learned rewards sustain PPO gains. Figure 3 shows that learning from multiple feedback types improves both downstream performance and robustness under policy optimization. All reward models initially improve over the starting policy, which achieves 38.6% GSM8K accuracy, but the magnitude and persistence of these gains differ. The scalar reward model and preference-only MELT reach peak accuracies of approximately 41.5%, whereas incorporating demonstrations raises the peak to 44.1%. Adding stopping feedback achieves a comparable peak of 44.0% and retains 43.3% accuracy at the final checkpoint.

Panels (b) and (c) reveal a corresponding difference under policy drift. For the scalar reward model, predicted return continues to increase with KL even after GSM8K accuracy begins to decline, indicating reward over-optimization. Preference-only MELT is more robust, maintaining approximately constant accuracy as KL increases, but further policy movement yields little additional improvement. In contrast, the models trained with multiple feedback types continue to translate larger policy updates into externally measured performance: [P][D][S] retains its gains beyond the target KL, while [P][D] declines only modestly at the largest divergences. These results suggest that complementary feedback produces reward signals that are both more effective for policy optimization and more robust as the optimized policy moves away from the training distribution.

6 Conclusion

We introduced a unified and scalable approach to learning dense rewards from diverse feedback. We use Bayesian inference to learn a shared reward model from demonstrations, preferences, and other feedback. Our experiments on mathematical reasoning and safety tasks show that complementary feedback can improve response ranking and sharpen error localization. In mathematical reasoning, the learned rewards also support higher policy accuracy and more sustained gains during PPO, suggesting greater robustness to reward over-optimization than the scalar reward baseline. These findings highlight learning jointly from diverse feedback as a promising direction for language model post-training. A natural next step is a comparison with conventional post-training pipelines across a broader range of model capabilities. Moreover, the uncertainty estimates offer further opportunities to guide feedback collection, including which feedback type to request, and to account for uncertainty in the learned rewards during policy optimization.

Limitations. We found that the strongest feedback combinations vary across tasks and benchmarks, and adding more feedback types does not always yield further gains. In safety, for example, the best three-type combinations outperform using all four types. This motivates further study of feedback selection. Our evaluation is limited to mathematical reasoning and safety tasks with a 3B-parameter backbone, including downstream policy optimization in mathematical reasoning. Extending these experiments to larger models and broader capabilities is an important next step.

Ethics Statement

This work studies reward learning from heterogeneous feedback for language model post-training. All experiments use publicly available datasets under their respective licenses; no new human data was collected, and no personal or identifying information is involved. The safety datasets contain prompts and responses describing harmful or unsafe behavior; we use this material solely to train and evaluate reward models that distinguish safe from unsafe content, and we do not release any generated harmful outputs. Reward models can encode biases present in their training data, and dense rewards may be exploited during policy optimization; we partially address the latter through length matching (Section B.1) and report failure cases and limitations openly. More broadly, the collection and use of user interaction data (edits, interruptions, ratings) for learning must be accompanied by appropriate transparency, consent, and governance mechanisms, so that users understand how their feedback is used and retain control over it. We view this as a prerequisite for deploying the approach with real user data.

AI Use Statement

We used generative AI tools, including ChatGPT (Astra, Sol) and Claude (Fable, Opus), for proposing and refining hypotheses, for providing feedback on research methodology and experiments, and assisting for implementing methods. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, generating synthetic datasets, cleaning and reformatting datasets, or interpreting results. Additionally, we used generative AI tools for creating and modifying scientific figures, drafting parts of the paper, and editing the paper to improve readability. We treated AI-assisted outputs as suggestions and reviewed, edited, and verified them before inclusion. AI-assisted code, including the method implementation, evaluation scripts, and plotting and data pipelines, was reviewed by the authors. All AI-drafted or AI-edited text was reviewed by all authors, and claims were verified against our experimental results and the cited sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility Statement

The model, feedback likelihoods, and objective are specified in Sections 2 to 4, with the full training procedure in Algorithm 1. All feedback is derived from public datasets; Section B gives exact sizes, preprocessing, and benchmark decontamination, and Section C.5 describes the error-localization benchmark. Sections C and D.2 list all reward model and PPO hyperparameters, seeds, and evaluation settings. Results are averaged over multiple seeds with standard errors, and ablations appear in Section E. Code is provided as supplementary material.

References

Raphaël Baur, Yannick Metz, Maria Gkoulta, Mennatallah El-Assady, Giorgia Ramponi, and Thomas Kleine Buening. MAVRL: Learning reward functions from multiple feedback types with amortized variational inference. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.15206.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. URL https://doi.org/10.2307/2334029.

Yuang Cai, Yuyu Yuan, Jinsheng Shi, and Qinhong Lin. Approximated variational Bayesian inverse reinforcement learning for large language model alignment. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, volume 39 of *AAAI'25/IAAI'25/EAAI'25*, pp. 23505–23513. AAAI Press, February 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i22.34519. URL https://doi.org/10.1609/aaai.v39i22.34519.

Alex James Chan and Mihaela van der Schaar. Scalable Bayesian inverse reinforcement learning. In *International Conference on Learning Representations 2021*, 2021. URL https://openreview.net/forum?id=4qR3coiNaIv.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4302–4310, Red Hook, NY, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Karel D'Oosterlinck, Winnie Xu, Chris Develder, Thomas Demeester, Amanpreet Singh, Christopher Potts, Douwe Kiela, and Shikib Mehri. Anchored preference optimization and contrastive revisions: Addressing underspecification in alignment. *Transactions of the Association for Computational Linguistics*, 13:442–460, 2025. doi: 10.1162/tacl_a_00748. URL https://aclanthology.org/2025.tacl-1.22/.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.01306.

Claudio Fanconi, Nicolás Astorga, and Mihaela van der Schaar. Learning Reasoning Rewards from Expert Demonstrations with Inverse Reinforcement Learning, May 2026. URL http://arxiv.org/abs/2510.01857. arXiv:2510.01857 [cs.AI].

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2023. URL https://zenodo.org/records/10256836.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2507.17746.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023. URL https://arxiv.org/abs/2305.14992.

Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic Preference Optimization without Reference Model. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.626. URL https://aclanthology.org/2024.emnlp-main.626/.

Shengyi Huang, Lewis Tunstall, Edward Beeching, Leandro von Werra, Omar Sanseviero, Kashif Rasul, and Thomas Wolf. Constitutional AI recipe. *Hugging Face Blog*, 2024.

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, 2026. URL https://arxiv.org/abs/2601.20802.

Hong Jun Jeon, Smitha Milli, and Anca D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.nips.cc/paper/2020/hash/2f10c1578a0706e06b6d7db6f0b4a6af-Abstract.html.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. BeaverTails: Towards improved safety alignment of LLM via a human-preference dataset. *arXiv preprint arXiv:2307.04657*, 2023. URL https://arxiv.org/abs/2307.04657.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. doi: 10.48550/arXiv.1312.6114.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2603.12273.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling, 2024. URL https://arxiv.org/abs/2403.13787.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath 1.5. Hugging Face repository, https://huggingface.co/datasets/AI-MO/NuminaMath-1.5, 2025. Project Numina.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation, 2025. URL https://arxiv.org/abs/2506.01937.

Dipendra Misra, Aldo Pacchiano, Ta-Chung Chi, and Ge Gao. Principled fine-tuning of llms from user-edits: A medley of preference, supervision, and reward, 2026. URL https://arxiv.org/abs/2601.19055.

NVIDIA. Nemotron-RL-Safety-v1. Hugging Face repository, https://huggingface.co/datasets/nvidia/Nemotron-RL-Safety-v1, 2025a.

NVIDIA. Nemotron-SFT-Safety-v1. Hugging Face repository, https://huggingface.co/datasets/nvidia/Nemotron-SFT-Safety-v1, 2025b.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf,

Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025.

Long Ouyang, Jeff 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*, 2022. URL https://arxiv.org/abs/2203.02155.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.18290.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to Q^*: Your language model is secretly a Q-function, 2024. URL https://arxiv.org/abs/2404.12358.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2586–2591, 2007. URL https://www.ijcai.org/Proceedings/07/Papers/416.pdf.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.01241.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback at scale, 2024. URL https://arxiv.org/abs/2303.16755.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. URL https://arxiv.org/abs/2211.14275.

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, 2023. URL https://arxiv.org/abs/2306.01693.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing, 2024. URL https://arxiv.org/abs/2406.08464.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. Advancing LLM reasoning generalists with preference trees, 2024. URL https://arxiv.org/abs/2404.02078.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. RRHF: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023. doi: 10.48550/arXiv.2304.05302. URL https://arxiv.org/abs/2304.05302.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008. URL https://cdn.aaai.org/AAAI/2008/AAAI08-227.pdf.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. URL https://arxiv.org/abs/1909.08593.

A Related Work

Learning from heterogeneous and rich feedback. Demonstrations, pairwise comparisons, and binary ratings are established sources of supervision for language models, typically entering training through separate objectives such as supervised fine-tuning, reward modeling or DPO, and KTO (Ouyang et al., 2022; Rafailov et al., 2023; Ethayarajh et al., 2024). Methods such as RRHF and ORPO combine supervised and comparative objectives into a single loss (Yuan et al., 2023; Hong et al., 2024), but remain limited to these two feedback types. Edits have also been used for policy learning: D'Oosterlinck et al. (2025) construct preference pairs from minimal LLM revisions, while Misra et al. (2026) learn from user edits through supervised fine-tuning, preference optimization, and reinforcement learning with edit-distance costs. A complementary line of work exploits textual feedback such as critiques and follow-up user messages, either to guide refinement and construct comparisons (Scheurer et al., 2024) or as targets for policy self-distillation, as in SDPO with environment feedback (Hübotter et al., 2026) and related methods learning from user interactions and critiques (Kleine Buening et al., 2026; Song et al., 2026). These approaches typically convert each signal into a fixed training objective, such as a preference pair, a supervised target, or a self-distillation target. We instead treat every signal as an observation of a single latent token-level reward, so that each type keeps its native granularity and all of them inform the same reward model. Several of our likelihoods reduce to familiar objectives in isolation: preferences to Bradley–Terry reward modeling, ratings to a KTO-style unary signal, and demonstrations to a Boltzmann-rational choice model whose negative log-likelihood is token-level cross-entropy. To our knowledge, user interruptions have not previously been modeled as a reward-learning signal for language models.

Probabilistic reward learning. Bayesian inverse reinforcement learning treats the reward as a latent variable inferred from observed behavior (Ramachandran & Amir, 2007), and reward-rational choice extends this view to diverse feedback types as observations of a shared reward (Jeon et al., 2020). AVRIL scales this inference with amortized variational inference, coupling the reward to an auxiliary action-value model through temporal consistency (Chan & van der Schaar, 2021). AVA applies this approach to language-model alignment from demonstrations and preferences (Cai et al., 2025), and MAVRL combines multiple feedback likelihoods in a shared variational posterior (Baur et al., 2026). We bring this perspective to modern language models as a unified framework for learning token-level rewards from a broad range of feedback types, each entering a shared objective through its own likelihood. Amortizing both the reward posterior and the action values over a shared pretrained transformer makes this inference practical at the scale of autoregressive generation.

Learning fine-grained reward signals. Fine-grained reward learning provides supervision within a response. Process reward models are trained on judgments of intermediate reasoning steps (Uesato et al., 2022; Lightman et al., 2024), and Fine-Grained RLHF uses localized annotations of response quality (Wu et al., 2023). Other work infers token-level rewards from a single feedback source, either from preferences through the implicit Q-function of DPO-trained models (Rafailov et al., 2024) or from demonstrations through adversarial IRL (Fanconi et al., 2026). Rather than requiring dedicated step- or span-level annotations or relying on a single feedback type, we study how several complementary modalities jointly determine token-level rewards, and evaluate the result on response ranking, error localization, and policy optimization.

B Dataset Construction and Preprocessing

Table 3 reports the sources, composition, and final sizes of the feedback datasets. We use comparable sizes within each domain where sufficient eligible examples are available.

Common preprocessing. We remove empty and duplicate examples and limit assistant responses to 2048 tokens (Llama-3.2-3B-Instruct tokenizer; both must comply in paired samples, overlength samples are dropped). To prevent train/validation leakage, all observations sharing a prompt or response are assigned to the same partition.

Domain	Feedback	Source(s) (share of total)	Train	Validation	Total
Math	P	Dolci-Instruct-DPO (Olmo et al., 2025) (28.6%), UltraInteract (Yuan et al., 2024) (40.1%), Magpie (Xu et al., 2024) (31.3%)	56,484	2,974	59,458
Math	D	NuminaMath-1.5 (Li et al., 2025) (50%), Dolci-Instruct-SFT (Olmo et al., 2025) (50%)	57,000	3,000	60,000
Math	S	PRM800K (Lightman et al., 2024)	57,000	3,000	60,000
Safety	P	Nemotron-RL-Safety-v1 (NVIDIA, 2025a)	40,000	2,000	42,000
Safety	D	Nemotron-SFT-Safety-v1 (NVIDIA, 2025b)	42,247	2,000	44,247
Safety	E	CAI Conversation Harmless (Huang et al., 2024)	15,300	827	16,127
Safety	R	BeaverTails (Ji et al., 2023)	40,000	2,000	42,000

Table 3: Feedback datasets after preprocessing and selection. Source shares refer to each row's combined total; a single source supplies the entire row.

B.1 Length Matching

We length-match preferences and binary ratings to reduce associations between response length and quality labels. Both likelihoods depend on cumulative reward, $G_R(x, y) = \sum_{t=1}^{T} R(s_t, a_t)$.

For preferences, let T^+ and T^- denote the preferred and non-preferred response lengths. We group pairs by their mean length, $\bar{T} = (T^+ + T^-)/2$, and absolute relative length difference, $|d|$, where $d = (T^+ - T^-)/(T^+ + T^-)$. Within each group, we retain equal numbers of pairs in which the preferred response is longer or shorter, subsampling the more frequent direction. Near-equal-length pairs ($|d| \leq 0.01$) are retained without matching. For binary ratings, we group responses by token length and retain equal numbers of positively and negatively labeled examples within each group.

B.2 Constructing Stops and Edits

Stopping feedback. We construct stopping feedback from the Phase 2 human step-level annotations of PRM800K (Lightman et al., 2024). For trajectories annotated as containing an error, we truncate the solution after the first step rated -1 and treat that step boundary as the stopping position. We retain only these stopped examples, yielding a subset of approximately 60,000 examples.

Edit feedback. We construct edit feedback from a Constitutional AI critique-and-revision dataset (Huang et al., 2024). We combine the provided training splits for training and the test splits for validation, extracting the original response, its critique, and the corresponding revised response for each example. To retain localized revisions rather than near-identical responses or complete rewrites, we filter examples by normalized word-level Levenshtein distance, $d_{\text{edit}}/\max(|y|, |y'|)$, retaining values between 0.05 and 0.90. We additionally remove examples with missing or very short responses and discard examples for which either response exceeds 2048 tokens under the Llama-3.2 tokenizer.

B.3 Benchmark Decontamination

We audit prompts from every math and safety reward-model feedback modality against the corresponding categories of RewardBench 1 (RB1) and RewardBench 2 (RB2). The audit includes training and validation splits; for multi-turn examples, we check every user turn. Counts refer to distinct benchmark prompts and use mutually exclusive match categories (exact, then embedded, then close).

Matching procedure. We collapse consecutive whitespace and strip leading and trailing whitespace. An *exact match* has identical text after this normalization; an *embedded match* occurs when a

benchmark prompt appears within a longer dataset prompt. These checks preserve case, punctuation, and mathematical notation.

To screen for similar wording, we lowercase prompts and split them into sequences of letters, digits, and underscores, discarding punctuation and mathematical operators. Let $G(x)$ be the set of overlapping token 5-grams in prompt x. For benchmark prompt b and dataset prompt t, we compute

$$c(b,t) = \frac{|G(b) \cap G(t)|}{|G(b)|}.$$

We flag b as a *close match* if any single dataset prompt contains at least 80% of its 5-grams. Prompts with fewer than five tokens receive only the exact and embedded checks.

We inspect every close match to determine whether the prompts pose the same math problem or safety request. In particular, we compare the constants, operators, and requested quantities in math problems, and the intent and target of safety requests. A close match is not necessarily a duplicate: discarding operators, for instance, can make different mathematical problems appear similar.

Benchmark category	Evaluation items	Exact	Embedded	Close
RB1 Math	447 prompts	0	0	14
RB2 Math	183 prompts	0	0	0
RB1 Safety	740 pairs	3	0	3
RB2 Safety	450 prompts	0	1	0

Table 4: Prompt overlap with the corresponding reward-model training and validation data. Exact counts identical prompts after whitespace normalization; Embedded counts additional benchmark prompts appearing within longer dataset prompts; Close counts additional prompts with 5-gram containment of at least 0.8. Columns are mutually exclusive and count distinct benchmark prompts. Close flags are assessed manually below. RB1 Safety contains 740 pairs over 738 unique prompts.

Mathematics. No RB1 or RB2 math prompt matches exactly or appears within a longer dataset prompt. The 14 RB1 close matches are different problems with similar wording or shared MATH templates. For example, a divisor changes from 9 to 8, the requested quantity changes from $\tan A$ to $\cos A$, or an addition becomes a subtraction. We find no close matches in RB2 Math.

Because the stop data and RB1 Math both derive from PRM800K, we also check their underlying problems and completions. We find that no stop example comes from the 500 held-out MATH problems in the PRM800K test split that supplies RB1 Math, and no stop completion is identical to an RB1 chosen or rejected response. Partial response matches are generic opening steps from different problems.

Safety. Three RB1 Safety prompts occur verbatim in training data, but none of their training responses is identical to an RB1 chosen or rejected response. We exclude the three evaluation pairs and report RB1 Safety on the remaining 737 pairs in Table 2. Recomputing every configuration and seed changes five-seed mean accuracy by between -0.09 and $+0.07$ percentage points.

Of the three additional RB1 close flags, two are differently worded requests with the same intent and one changes the request's target. RB2 Safety has no exact or close matches. Its sole embedded match is a one-word prompt occurring within a much longer training request.

C Implementation and Training Details

C.1 Shared Initialization and Model Architecture

Supervised initialization. All policies and reward models start from a shared checkpoint obtained by supervised fine-tuning the base Llama 3.2 3B model on approximately 20k examples from the general chat, instruction-following, and reasoning portions of Dolci-Instruct-SFT (Olmo et al., 2025). This stage adapts the model to the Llama 3 chat template, including role headers, turn structure, and end-of-turn tokens. The shared checkpoint controls for initialization across reward-learning configurations and subsequent PPO runs.

Reward and action-value heads. The reward and action-value heads share the transformer backbone of this SFT checkpoint (hidden size 3,072; vocabulary size $|\mathcal{V}| = 128{,}256$). Both heads are linear projections of its final-layer hidden states. For each generated token a_t, the reward head predicts the mean and log-variance of a Gaussian token-reward distribution:

$$\big(\mu_\theta(s_t, a_t), \log \sigma^2_\theta(s_t, a_t)\big) = W_r h(s_t, a_t) + b_r.$$

We use PyTorch's default linear-layer initialization for this head. The action-value head predicts a value for every possible next token from the hidden state of the preceding prefix:

$$Q_\phi(s_t, \cdot) = W_Q h(s_t) + b_Q.$$

We initialize $W_Q = W_{\mathrm{LM}}/\beta_{\mathrm{demo}}$ from the SFT checkpoint's language-model head and set $b_Q = 0$. The backbone and both heads are trained jointly.

C.2 Reward-Model Training

Table 5 summarizes the shared optimization settings, and Table 6 gives the loss coefficients. Each optimizer step draws one minibatch from every active feedback modality. Early stopping and checkpoint selection use validation loss macro-averaged across active modalities. We train each configuration with five seeds, holding its hyperparameters fixed across seeds.

Setting	Value
Optimizer; learning rate	AdamW; 10^{-5}
Learning-rate schedule	Linear decay; warmup fraction 0.01
Batch size per GPU; accumulation	8; 2 steps
Total batch size	64
Maximum sequence length	2048 tokens
Maximum training duration	6 epochs
Early stopping	Patience 10 evaluations; improvement threshold 0
Validation criterion	Macro-average loss over active modalities
Precision; training seeds	bfloat16; 42–46

Table 5: Common reward-model training settings.

Coefficient	Value
Prior KL coefficient λ_{KL}	0.1 by default; 0.01 in the prior ablation
TD coefficient λ_{TD}	0.1 if any active feedback likelihood depends on Q_ϕ; 0 otherwise

Table 6: Reward-model loss coefficients.

C.3 Observation-Model Parameters

Table 7 reports the observation-model parameters for the likelihoods in Section 3. We use the same settings across seeds and modality combinations within each domain. The sensitivity coefficients act on different quantities and therefore need not have comparable numerical values: preferences and ratings depend on cumulative token rewards, edits on individual token rewards, and demonstrations on action-value differences. The corrective-demonstration factor of the edit likelihood uses the same β_{demo} as standalone demonstrations. Rating and edit thresholds are fixed at zero.

C.4 Scalar Baseline and Computational Cost

Scalar baseline. The scalar baseline uses the same SFT checkpoint with a linear score head applied to the last non-padding token. We train it on the same preference data using the Bradley–Terry loss, $-\log \sigma\big(R_\psi(x, y^+) - R_\psi(x, y^-)\big)$, and the optimization settings in Table 5. Validation loss determines early stopping and checkpoint selection.

Modality	Domain	Parameter settings
Preferences	Math, Safety	$\beta_{\text{pref}} = 0.01$
Ratings	Safety	$\beta_{\text{rate}} = 0.01$, $\tau_{\text{rate}} = 0$
Demonstrations	Math, Safety	$\beta_{\text{demo}} = 1.0$
Edits	Safety	$\beta_{\text{edit}} = 1.0$, $\tau_{\text{edit}} = 0$, $\beta_{\text{demo}} = 1.0$
Stops	Math	$\lambda_{\text{stop}} = 0.01$, $\gamma_{\text{s}} = 0.98$, $\beta_{\text{s}} = 5.0$

Table 7: Observation-model parameter settings. All parameters are held fixed during reward inference. Sensitivity and smoothing parameters are selected using held-out validation data; rating and edit thresholds are fixed at zero.

Training overhead. Compared with a scalar Bradley–Terry reward model on the same backbone, MELT adds one head and two auxiliary losses. Its reward head replaces the scalar score head and differs only in having two outputs, a mean and a log-variance, instead of one. The additional action-value head is a full $3{,}072 \times |\mathcal{V}|$ projection, which adds 394M parameters (about 12% of the 3.2B backbone), together with their gradients and optimizer states. Its full-vocabulary output is needed only for the demonstration likelihoods, that is, for demonstrations and the corrective part of edits. There it costs the same as a standard language-modeling loss: about 0.8 GFLOPs per token, or roughly 14% of a backbone forward pass. For preferences, ratings and stops, the TD loss needs Q_ϕ only at the taken actions, so it is computed as a dot product with the corresponding rows of the head, at negligible cost. The prior KL and TD losses are elementwise over tokens. Per example, a MELT preference model therefore costs essentially the same as the scalar baseline: one backbone pass per response. The main cost of multi-feedback training comes from the data rather than the model. Each optimizer step draws one minibatch from every active modality, so a step with k modalities uses about k times the backbone compute of a single-modality step. At deployment the overhead disappears. Scoring uses only the backbone and the reward head, the action-value head can be discarded, and one forward pass gives the reward of every token, at the same cost at which a scalar model gives one score per response.

Compute. All experiments were run on NVIDIA GH200 GPUs. Training one reward model takes 5–24 GPU-hours, and one 200-step PPO run takes 4–9 GPU-hours. Reproducing all reported results, including appendix ablations and evaluation, requires approximately 2.9k GPU-hours.

C.5 Training Procedure

Each optimizer step draws a minibatch from every active feedback modality. Datasets are shuffled independently, and smaller datasets are cycled when exhausted. An epoch corresponds to one pass over the largest active dataset. To account for the repeated use of smaller datasets, we weight each modality by $w_m = N_m / \max_{m'} N_{m'}$, where N_m is its number of training examples after filtering. The objective is

$$\mathcal{L} = \sum_m w_m \mathcal{L}_{\text{NLL}}^{(m)} + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}} + \lambda_{\text{TD}} \mathcal{L}_{\text{TD}}^{\text{loss}}.$$

Preference and rating likelihoods are averaged over examples, demonstration likelihoods over response tokens, and stopping likelihoods over trajectories. The edit likelihood combines token-averaged terms for the original and revised responses. The KL and TD terms are averaged over reward tokens using the same modality weights.

TD-target alignment. Since the action-value head is initialized from language-model logits, its temporal differences initially differ in scale from the predicted rewards. Under the Gaussian TD-consistency likelihood, large residuals from this scale mismatch can encourage the reward posterior to increase its predicted variance. We therefore standardize the temporal differences within each minibatch and rescale them to the mean and standard deviation of the corresponding predicted reward means:

$$\widehat{\delta}_{\phi,t} = \frac{\delta_{\phi,t} - \mu_{\delta,\mathcal{B}}}{\sigma_{\delta,\mathcal{B}} + \varepsilon} \, \sigma_{\mu,\mathcal{B}} + \mu_{\mu,\mathcal{B}}.$$

The TD-consistency likelihood uses $\widehat{\delta}_{\phi,t}$; the no-alignment ablation uses $\delta_{\phi,t}$ directly.

Algorithm 1 summarizes optimization of Eq. (4).

Algorithm 1 MELT

Require: Feedback datasets $\{\mathcal{D}^{(m)}\}_{m=1}^{M}$; coefficients $\lambda_{\mathrm{KL}}, \lambda_{\mathrm{TD}}$; learning rate η
Initialize the shared backbone from the SFT checkpoint; attach a reward head for q_θ and initialize Q_ϕ from the LM head
while not converged **do**
  Sample feedback minibatches $\mathcal{B} = \{\mathcal{B}^{(m)}\}_{m=1}^{M}$
  Predict token-reward parameters $(\mu, \sigma) \leftarrow q_\theta(\mathcal{B})$ and action values $Q \leftarrow Q_\phi(\mathcal{B})$
  Sample rewards $R \leftarrow \mu + \sigma \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$
  Compute $\mathcal{L}_{\mathrm{NLL}}^{(m)} \leftarrow -\log p^{(m)}(\mathcal{B}^{(m)} \mid R, Q)$ for $m = 1, \ldots, M$
  $\mathcal{L}_{\mathrm{KL}} \leftarrow D_{\mathrm{KL}}(q_\theta \,\|\, \mathcal{N}(0, I))$
  $\mathcal{L}_{\mathrm{TD}}^{\mathrm{loss}} \leftarrow 0$
  if $\lambda_{\mathrm{TD}} > 0$ **then**
    Compute $\delta_t \leftarrow Q_\phi(s_t, a_t) - Q_\phi(s_{t+1}, a_{t+1})$, with zero continuation value at termination
    If alignment is enabled, set $\widehat{\delta}_t \leftarrow \dfrac{\delta_t - \mu_{\delta,\mathcal{B}}}{\sigma_{\delta,\mathcal{B}} + \varepsilon}\,\sigma_{\mu,\mathcal{B}} + \mu_{\mu,\mathcal{B}}$; otherwise set $\widehat{\delta}_t \leftarrow \delta_t$
    $\mathcal{L}_{\mathrm{TD}}^{\mathrm{loss}} \leftarrow -\mathcal{L}_{\mathrm{TD}}(\theta, \phi; \widehat{\delta})$ (Eq. 3)
  end if
  $\mathcal{L} \leftarrow \sum_{m=1}^{M} \mathcal{L}_{\mathrm{NLL}}^{(m)} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{TD}}\mathcal{L}_{\mathrm{TD}}^{\mathrm{loss}}$
  $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla_{\theta,\phi}\mathcal{L}$
end while
return q_θ, Q_ϕ

D Evaluation Protocols

D.1 Controlled Error Localization

Dataset construction. We build a controlled error-localization benchmark by adding exactly one arithmetic error to otherwise correct GSM8K reference solutions (Cobbe et al., 2021). We deduplicate problems by normalized question text. For each solution we remove the calculator annotations (`<<...>>`) from the reasoning and keep the final-answer line (`#### <answer>`) exactly as it is.

We use pattern matching to find explicit, unparenthesized arithmetic equalities in the reasoning: integers or decimals joined by $+$, $-$, $\times$, or $/$. Each candidate is checked with exact rational arithmetic, and we discard any expression that is ambiguous, contains variables, or does not hold exactly.

We corrupt a verified equality in one of two ways. A *result corruption* shifts the displayed result by ± 1 or ± 2 in its last displayed digit, keeping its sign and character width. An *operator swap* replaces one binary operator with a different one that makes the equation false. We re-verify every corrupted equation under exact arithmetic to confirm that it is false. All surrounding text and the original final answer remain unchanged.

For each example we record the character span of the corrupted text and the span of the equation's result within the annotation-free solution. The corrupted span defines the evaluated localization target; the result span is retained as metadata. Examples are sampled with a fixed random seed, split evenly between the two corruption types (250 each, 500 in total), and any question that overlaps the reward-model training data is excluded.

Localization protocol. We condition each corrupted solution on its question and score it with the reward model, using the same tokenization as in training: the chat-templated prompt followed by the tokenized response. This yields a per-token mean reward μ_t. The ground-truth target is the first response token whose character span overlaps the corrupted span, found through the tokenizer's offset mapping. For operator swaps this is the swapped operator, not the equation's result.

The prediction is the token with the largest one-step reward drop, $\hat{t} = \arg\max_t \Delta_t$ with $\Delta_t = \max(0, \mu_{t-1} - \mu_t)$, and ties go to the earliest position. The argmax runs only over reasoning tokens. We exclude prompt and template tokens, the final-answer line, and the first reasoning token, whose Δ_t reflects the jump from prompt to response rather than a reasoning error. A trace in which the reward never drops within the reasoning region counts as a miss and stays in the denominator.

We report Hit@k for $k \in \{1, 5\}$: the fraction of examples where $|\hat{t} - t^\star| \leq k$ tokens. We compare against a length-aware chance baseline, the expected Hit@k of choosing a candidate token uniformly at random, averaged over examples.

D.2 Policy Optimization

Setup. We use PPO (Schulman et al., 2017) to evaluate whether the learned reward models remain useful under policy optimization. The policy and frozen reference policy are initialized from the same conversational SFT checkpoint used to initialize the reward models (Appendix C.1), and the critic is initialized from the corresponding reward model. Across reward-model variants, we keep the policy initialization, prompt distribution, PPO objective, and optimization hyperparameters fixed. The task reward and corresponding critic initialization vary with the reward model. Prompts are deduplicated GSM8K training questions (Cobbe et al., 2021), wrapped in the same chat template used at evaluation.

Rewards. For the scalar Bradley–Terry baseline (Bradley & Terry, 1952), the response score is supplied only at the final generated token, $r_t^{\text{scalar}} = \mathbf{1}\{t = T\} R_\psi(x, y)$. For MELT, we instead sample a reward at every generated position, $r_t^{\text{dense}} \sim \mathcal{N}(\mu_\theta(s_t, a_t), \sigma_\theta^2(s_t, a_t))$. To place the resulting returns on a comparable scale, we divide task rewards by the standard deviation of the summed sequence reward under the initial policy. This statistic is estimated once from 10 rollout batches before training and then held fixed. Reward scaling is applied only during policy optimization and does not modify the reward-learning objective. Responses that terminate without a stop token receive an additional penalty of 1. We use an adaptive KL penalty with the proportional controller of Ziegler et al. (2019).

Evaluation. We evaluate checkpoints throughout training on the 1,319-problem GSM8K test set using `lm-evaluation-harness` (Gao et al., 2023). Evaluation uses 8-shot chain-of-thought prompting (`gsm8k_cot_llama`), with few-shot examples presented as chat turns, greedy decoding with at most 512 new tokens, and flexible exact match on the final answer.

Results in the main PPO figure average eight runs: two reward-model seeds and four PPO seeds per reward model.

E Additional Reward-Model Results

We control the total feedback budget in mathematical reasoning and examine TD-target alignment and prior regularization in both domains. Scores are mean ± standard error (%) across five training seeds. RB1 and RB2 denote the domain-specific categories described in Section 5.3. Feedback symbols follow the main text.

E.1 Matched Feedback Budget

Combined models in Table 2 use the full dataset of each modality and therefore observe more feedback in total than single-modality models. To separate the effect of combining feedback types from that of additional data, we train mathematical-reasoning models on a fixed total budget of 57k examples, approximately matching the preference-only dataset size, divided evenly across the active modalities (28.5k per modality for pairs, 19k for three modalities). All other settings follow Section 5.1. This control matches feedback examples, not response counts, token counts, or compute.

At a matched total feedback budget, heterogeneous feedback can replace a substantial portion of preference data while maintaining competitive reward-model performance. Models using only half or one-third as many preferences achieve higher mean accuracy than the preference-only model on RB1, while remaining close to it on RB2. In particular, P D gains 9.22 points on RB1 with half the preference data and differs by less than one point from its full-data counterpart in Table 2 for RB1.

Complete substitution is more dependent on the evaluation setting. D S uses no preferences and achieves the highest RB1 score, but its RB2 score is substantially lower. These results suggest that other feedback types can replace much of the preference supervision; retaining some preferences provides more consistent performance across the two benchmarks.

Hyperparameter	Value
Policy initialization	Conversational SFT checkpoint
Prompt dataset	Deduplicated GSM8K train
Prompts / PPO step	32
PPO updates	200
Total episodes	6,400
Maximum response length	1024
Generation temperature	1.0
Top-p	1.0
Reward-scale estimate	10 initial rollout batches
Discount γ	1.0
GAE λ	0.95
PPO epochs / batch	2
Minibatches / epoch	1
Policy clip range	0.1
Value clip range	0.2
Value-loss coefficient	1.0
Policy learning rate	1×10^{-6}
Critic learning rate	2×10^{-6}
LR schedule	Constant after 7.5% warmup
Gradient-norm clipping	1.0
Weight decay	0
KL controller	Adaptive proportional
Initial KL coefficient	0.05
Target KL	24 nats / response
KL horizon	500 episodes
Missing-stop penalty	1.0
Precision	`bf16`

Table 8: PPO hyperparameters shared across reward-model variants.

Feedback	Examples per modality	RB1	RB2
P	56,484	76.87 ± 3.13	53.33 ± 0.61
P D	28,500	86.09 ± 2.00	51.15 ± 0.78
P S	28,500	79.78 ± 3.60	52.35 ± 1.86
D S	28,500	93.47 ± 1.17	30.16 ± 2.57
P D S	19,000	83.27 ± 1.33	52.35 ± 1.76

Table 9: Reward-model performance in mathematical reasoning with the combined-model feedback budget fixed at 57k examples. The preference-only reference uses 56,484 examples. Scores are mean $\pm$ standard error (%) over five seeds. P: preferences; D: demonstrations; S: stops.

E.2 TD-Target Alignment and Prior Regularization

We examine two choices in mathematical reasoning and safety: removing TD-target alignment while retaining $\lambda_{\mathrm{KL}} = 0.1$, and reducing λ_{KL} to 0.01 while retaining alignment. Tables 10 and 11 report the respective results. The evaluation subsets follow Section 5.3, using RB1 (Lambert et al., 2024) and RB2 (Malik et al., 2025). Scalar baselines are repeated for reference.

TD-target alignment. Removing alignment has mixed effects: math RB1 accuracy for PDS decreases from 94.27% to 88.68%, while PD changes from 85.19% to 86.71%. Configurations without a TD term reuse their default results.

Mathematical reasoning				Safety			
Model	Feedback	RB1	RB2	Model	Feedback	RB1	RB2
Scalar RM	P	74.45 ± 3.81	54.86 ± 1.65	Scalar RM	P	59.97 ± 2.61	50.13 ± 2.50
MELT	P	76.87 ± 3.13	53.33 ± 0.61	MELT	P	71.78 ± 0.98	51.87 ± 1.58
MELT	D	19.78 ± 4.24	19.13 ± 0.60	MELT	D	51.62 ± 0.88	34.22 ± 1.45
MELT	S	66.13 ± 1.56	35.19 ± 1.85	MELT	E	56.19 ± 0.95	25.56 ± 1.24
				MELT	R	69.99 ± 0.59	48.36 ± 1.27
MELT	PD	86.71 ± 1.54	**55.52** ± 1.49	MELT	PD	75.00 ± 0.77	56.13 ± 1.37
MELT	PS	89.31 ± 2.94	54.64 ± 1.36	MELT	PE	69.16 ± 1.69	53.96 ± 2.26
MELT	DS	**94.77** ± 1.75	26.34 ± 1.86	MELT	PR	73.05 ± 0.53	62.22 ± 2.19
				MELT	DR	70.97 ± 0.72	49.73 ± 2.12
MELT	PDS	88.68 ± 1.61	53.88 ± 2.22	MELT	PDE	**75.05** ± 1.18	57.24 ± 1.65
				MELT	PDR	73.76 ± 0.41	**64.44** ± 1.55
				MELT	PDER	74.14 ± 1.14	52.27 ± 1.94

Table 10: Reward-model performance without TD-target alignment ($\lambda_{\mathrm{KL}} = 0.1$). Highest column means are bold; scalar baselines are repeated for reference.

Prior regularization. With alignment retained, reducing λ_{KL} to 0.01 improves mean math RB2 accuracy for every configuration containing preferences, while RB1 effects vary. In safety, PD improves on both benchmarks and PR reaches 64.72% on RB2, but several other configurations decline. Neither change improves performance uniformly.

Mathematical reasoning				Safety			
Model	Feedback	RB1	RB2	Model	Feedback	RB1	RB2
Scalar RM	P	74.45 ± 3.81	54.86 ± 1.65	Scalar RM	P	59.97 ± 2.61	50.13 ± 2.50
MELT	P	75.30 ± 1.62	54.75 ± 1.16	MELT	P	70.24 ± 0.97	52.36 ± 1.53
MELT	D	26.17 ± 15.45	29.51 ± 0.96	MELT	D	43.68 ± 4.12	24.04 ± 1.58
MELT	S	88.55 ± 2.16	33.77 ± 1.18	MELT	E	42.86 ± 0.89	22.53 ± 0.21
				MELT	R	71.76 ± 0.69	53.64 ± 3.22
MELT	PD	85.95 ± 1.32	**55.96** ± 1.35	MELT	PD	**79.11** ± 0.66	59.82 ± 0.85
MELT	PS	86.76 ± 3.31	55.74 ± 1.41	MELT	PE	70.92 ± 0.52	54.13 ± 1.99
MELT	DS	**96.24** ± 0.68	28.52 ± 1.77	MELT	PR	70.84 ± 1.38	**64.72** ± 2.61
				MELT	DR	70.46 ± 0.83	46.40 ± 2.14
MELT	PDS	90.56 ± 1.85	55.08 ± 1.52	MELT	PDE	71.81 ± 0.82	58.36 ± 1.59
				MELT	PDR	74.73 ± 0.40	61.29 ± 1.55
				MELT	PDER	72.68 ± 0.78	51.91 ± 1.15

Table 11: Reward-model performance with TD-target alignment and $\lambda_{\mathrm{KL}} = 0.01$. Highest column means are bold; scalar baselines are repeated for reference.