BEYOND FIXED RULES: BEST-OF-N ALIGNMENT WITH LEARNED SET-WISE SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Best-of-N, which generates N candidate responses from an LLM and selects the one with the highest reward-model score, is widely used for inference-time alignment but is susceptible to reward hacking as N grows. Some recent work seeks to mitigate this problem by designing proxy-reward correction rules based on signals over the N candidate responses. However, these methods encode predefined assumptions about reward-model errors and the rigidity of such fixed-form rules may limit their ability to capture reward-model error patterns that vary across inference-time conditions, making it difficult to reliably distinguish reward-hacked responses from genuinely high-quality ones. To move beyond fixed rules under varying inference-time conditions, we propose LESS, a LEarned Set-wise Selection framework for Best-of-N alignment, which learns a tailored selection strategy from the corresponding data to select the best response. It represents each candidate using reward and representation features, then employs a set transformer encoder to capture interactions among the N candidates and assign each response a selection logit, returning the highest-scoring candidate as the final output. The set-based design naturally accommodates varying candidate-set sizes. Extensive experiments across diverse benchmarks and varying values of N demonstrate the effectiveness of the proposed approach.

1 INTRODUCTION

Inference-time scaling has emerged as a powerful paradigm for improving the capabilities of large language models (LLMs) by allocating additional computation during inference (Zhang et al., 2025a; Guo et al., 2025). One important application of this paradigm is inference-time alignment, which seeks to steer model behavior toward human preferences such as correctness and helpfulness (Pan et al., 2025; Zhang et al., 2025c). Among inference-time alignment methods, Best-of-N is one of the most widely used approaches (Brown et al., 2024; Beirami et al., 2025). For each prompt, Best-of-N generates N candidate responses and selects the candidate with the highest reward score.

While attractive in its simplicity, Best-of-N is known to suffer from reward hacking as N increases (Gao et al., 2023; Khalaf et al., 2025). A growing body of work seeks to mitigate this problem by designing proxy-reward correction rules based on signals over the N candidate responses. Some methods use the candidate rewards distribution to regulate the aggressiveness of Best-of-N selection. Verdun et al. (2025) replace hard maximization with reward-weighted sampling, Huang et al. (2025) introduce pessimism through regularized rejection sampling, and Hsu et al. (2026) adapt the selection strategy according to the tail behavior of the candidate rewards. Other methods use the representations of the N candidates to estimate their reliability and adjust their reward scores. Jinnai et al. (2025) favor responses with high average similarity to the other candidates, whereas Yu et al. (2026) penalize candidates that appear out of distribution in representation space.

Although these methods help mitigate reward hacking in Best-of-N, they still have notable limitations in practice. On the one hand, these methods rely on manually designed proxy-reward correction rules that encode predefined assumptions about reward-model errors. The rigidity of such fixed-form rules may limit their ability to capture reward-model error patterns that vary across inference-time conditions or deviate from predefined assumptions, making it difficult to reliably distinguish reward-hacked responses from genuinely high-quality ones. On the other hand, their practical implementations may involve heuristic design choices, while the strength of the correction

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.

is typically controlled by manually tuned hyperparameters. This introduces additional tuning effort and may reduce performance stability in real-world settings.

To address these limitations, we propose LESS, a LEarned Set-wise Selection framework for Best-of-N alignment. Rather than relying on fixed rules, LESS learns a tailored selection strategy directly from the corresponding data to select the best response among the N candidates. It represents each candidate response using proxy-reward features and the prompt-response embedding, then employs a set transformer encoder to capture interactions among N candidates. The model assigns a selection logit to each response and is trained with a set-level objective tailored to final response selection, with the highest-scoring candidate chosen as the output. The set-based design naturally accommodates varying candidate-set sizes, providing a scalable and data-driven approach as inference-time compute increases. Our contributions can be summarized as follows:

- We propose LESS, a learned set-wise selection framework for Best-of-N alignment that learns a tailored selection strategy from the corresponding data to select the best response among the N candidates, moving beyond fixed rules under varying inference-time conditions.
- By learning the selection strategy directly from data, LESS eliminates the need for correction-strength hyperparameters, reducing setting-specific manual tuning.
- Extensive experiments across diverse benchmarks, different reward models, and varying values of N demonstrate the effectiveness of LESS.

2 Related Work

Best-of-N alignment generates N candidate responses and selects the highest-scoring according to a reward model. This technique has proven effective for preferences alignment (Stiennon et al., 2020; Barreto et al., 2025), mathematical reasoning (Cobbe et al., 2021; Lightman et al., 2024) and coding (Li et al., 2022; Wang et al., 2025). While attractive in its simplicity, Best-of-N is known to suffer from reward hacking, or reward overoptimization, as N increases (Gao et al., 2023; Khalaf et al., 2025). Specifically, the proxy reward of the selected response typically increases with N, while its ground-truth reward does not improve correspondingly and may even decline.

One line of work mitigates reward hacking in Best-of-N by improving reward models through tailored objectives, regularization techniques, or higher-quality preference data. Yang et al. (2024) regularize hidden representations with language-modeling objectives, while Hong et al. (2025) propose batch-wise sum-to-zero regularization. Li et al. (2025) introduce adaptive optimal-transport based margins, and Duan et al. (2026) learn sparse non-negative reward factors. Miao et al. (2024) and Li et al. (2026) suppress spurious signals with information-theoretic objectives, whereas Liu et al. (2025b) remove context-free artifacts through data augmentation. From a data-centric perspective, Liu et al. (2024) curate compact, high-quality preference mixtures, while Liu et al. (2025a) scale preference curation through human-AI collaboration. A separate line mitigates reward hacking through reward-model aggregation. Coste et al. (2024) apply conservative optimization over ensembled reward predictions, while Eisenstein et al. (2024) average predictions from diversely pretrained reward models. In contrast, Rame et al. (2024) directly average fine-tuned reward-model weights. Despite their effectiveness, training strong individual reward models and constructing robust reward-model ensembles often require substantial amounts of high-quality preference data and considerable computational resources.

A complementary line of work instead keeps the reward model fixed and improves Best-of-N selection by exploiting information over the N candidate responses. One class of methods uses candidate rewards to moderate selection strength. Verdun et al. (2025) replace hard maximization with reward-weighted sampling, Huang et al. (2025) use pessimistic regularized rejection sampling, and Hsu et al. (2026) adapt selection to the reward tails. Khalaf et al. (2025) further formalize inference-time hacking as a winner's curse and hedge against it with tuned Poisson-randomized selection. Another class uses candidate representations to assess reliability. Jinnai et al. (2025) favor candidates near the representation centroid, Ichihara et al. (2025) extend similarity regularization to stochastic Best-of-N, and Yu et al. (2026) penalize representation-space outliers. However, these methods rely on fixed-form correction rules and manually tuned hyperparameters, limiting their practicality under varying inference-time condition.

3 PROPOSED METHOD

3.1 PRELIMINARIES

We study Best-of-N, an inference-time alignment method in which an LLM generates N candidate responses and a selection rule chooses one of them as the final output. Consider an LLM π, where $\pi : \mathcal{X} \to \Delta(\mathcal{Y})$ maps each prompt $x \in \mathcal{X}$ to a distribution over responses $y \in \mathcal{Y}$. Given a prompt x, we first samples N candidate responses independently from $\pi(\cdot \mid x)$ and denote the responses by the indexed collection $\boldsymbol{y}(x) = [y_1, \ldots, y_N]$, where $y_i \sim \pi(\cdot \mid x)$ for $i \in [N]$. The selection of Best-of-N is typically guided by a proxy reward model $\hat{r} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assigns a scalar reward $\hat{r}(x, y)$ to each prompt-response pair. For $\boldsymbol{y}(x)$, the corresponding proxy rewards collection is given by $\hat{\boldsymbol{r}}(x, \boldsymbol{y}(x)) = [\hat{r}(x, y_1), \ldots, \hat{r}(x, y_N)]$. The standard Best-of-$N$ selects the response with the largest proxy reward.

However, the proxy rewards may become miscalibrated for Best-of-N as N grows. Many existing reward-calibration methods address this by introducing a calibration function f that maps the prompt, response collection, and proxy rewards to calibrated scores:

$$\tilde{\boldsymbol{r}}(x, \boldsymbol{y}(x)) = f(x, \boldsymbol{y}(x), \hat{\boldsymbol{r}}(x, \boldsymbol{y}(x))) = [\tilde{r}(x, y_1), \ldots, \tilde{r}(x, y_N)]\,. \tag{1}$$

The response with the highest calibrated score is selected, with its index given by $i_f(x, \boldsymbol{y}(x)) = \arg\max_{i \in [N]} \tilde{r}(x, y_i)$. Let $r^\star : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ denote the ground-truth reward function, which measures the actual quality of a response with respect to the desired alignment criterion. The objective of f is to maximize the probability of selecting a response from the candidate collection $\boldsymbol{y}(x)$ with the highest ground-truth reward:

$$f^\star \in \arg\max_f \mathbb{E}_{x, \boldsymbol{y}(x)} \left[\mathbf{1} \left\{ r^\star\big(x, y_{i_f(x, \boldsymbol{y}(x))}\big) = \max_{j \in [N]} r^\star(x, y_j) \right\} \right]. \tag{2}$$

3.2 OVERVIEW

Our framework learns a tailored, data-driven selection strategy for Best-of-N alignment. Given a prompt and its candidate responses, LESS first characterizes each candidate using reward-statistic features derived from the proxy rewards and a semantic representation produced by an embedding model. The two types of features are then combined to form candidate representations, which are jointly processed by a set transformer encoder to capture interactions among responses in the same candidate collection. Finally, a shared prediction head assigns a calibrated selection score to each candidate, and the response with the highest score is selected as the final output. The framework is trained with a set-level objective that encourages candidates with the highest ground-truth reward in each collection to receive the largest calibrated scores. In this way, LESS learns to tailor the selection strategy to different inference-time conditions from the corresponding data, rather than relying on fixed-form correction rules. The overall framework of LESS is illustrated in Figure 5.

3.3 THE LESS FRAMEWORK

We instantiate the calibration function f introduced in the preliminaries as a neural network θ. Given a prompt x, its candidate response collection $\boldsymbol{y}(x)$, and the corresponding proxy reward vector $\hat{\boldsymbol{r}}(x, \boldsymbol{y}(x))$, LESS constructs a representation for each candidate, jointly models the candidate collection, and outputs a calibrated selection score for every response.

3.3.1 SET-WISE SELECTION.

For each candidate response y_i, we construct a four-dimensional reward-statistic feature vector

$$\boldsymbol{a}_i = \left[\hat{r}(x, y_i), \max_{j \in [N]} \hat{r}(x, y_j) - \hat{r}(x, y_i), \mathrm{rank}(\hat{r}(x, y_i))\,/N, \log N \right] \in \mathbb{R}^4, \tag{3}$$

where $\mathrm{rank}(\hat{r}(x, y_i))$ denotes the rank of the proxy reward assigned to y_i among the N candidates. The four components respectively capture the proxy reward, the gap from the highest-reward candidate, the relative rank, and the candidate collection size. In addition to reward statistics, we use an embedding model E_{emb} to encode the semantic information of each prompt–response pair:

$$\boldsymbol{e}_i = E_{\mathrm{emb}}(x, y_i) \in \mathbb{R}^{d_e}. \tag{4}$$

The embedding $\boldsymbol{e}_i$ provides candidate-specific semantic information that cannot be fully represented by scalar proxy rewards. We then transform the two types of features using separate learnable mappings and concatenate their outputs:

$$\boldsymbol{u}_i = \mathrm{Concat}\left(g_a(\boldsymbol{a}_i), g_e(\boldsymbol{e}_i)\right) \in \mathbb{R}^d. \tag{5}$$

where $g_a : \mathbb{R}^4 \to \mathbb{R}^{d_r}$ and $g_e : \mathbb{R}^{d_e} \to \mathbb{R}^{d_s}$ are learnable single-layer linear projections for the reward-statistic and semantic features, respectively, and $d = d_r + d_s$. The resulting candidate representations are arranged into $\mathbf{U}(x, \boldsymbol{y}(x)) = [\boldsymbol{u}_1; \ldots; \boldsymbol{u}_N] \in \mathbb{R}^{N \times d}$. We use the encoder of the set transformer (Lee et al., 2019), consisting of stacked set attention blocks (SABs), to jointly process the candidate representations as a set:

$$\mathbf{C}(x, \boldsymbol{y}(x)) = g_{\mathrm{set}}(\mathbf{U}(x, \boldsymbol{y}(x))) = [\boldsymbol{c}_1; \ldots; \boldsymbol{c}_N] \in \mathbb{R}^{N \times d}, \tag{6}$$

where $\boldsymbol{c}_i$ is the representation associated with candidate y_i, contextualized by the entire candidate set. By stacking SABs, the encoder learns to capture pairwise and higher-order interactions among the candidates, which are encoded in the resulting contextualized representations. A shared linear prediction head g_o maps each contextualized representation $\boldsymbol{c}_i$ to a scalar calibrated score:

$$\tilde{r}_\theta(x, y_i) = g_o(\boldsymbol{c}_i). \tag{7}$$

The complete calibrated score vector is therefore $\tilde{\boldsymbol{r}}_\theta(x, \boldsymbol{y}(x)) = [\tilde{r}_\theta(x, y_1), \ldots, \tilde{r}_\theta(x, y_N)] \in \mathbb{R}^N$. Because both the SAB-based encoder and the shared prediction head operate independently of the set size, the same model can be applied to candidate sets with varying numbers of elements N. At inference time, LESS selects and returns the final response $y_{i_\theta(x, \boldsymbol{y}(x))}$, where $i_\theta(x, \boldsymbol{y}(x)) = \arg\max_{i \in [N]} \tilde{r}_\theta(x, y_i)$.

3.3.2 Learning Objective.

The learnable parameters θ of the calibration function f are associated with the reward-feature projection g_a, the embedding projection g_e, the set-wise interaction module g_{set}, and the shared prediction head g_o. We jointly optimize these learnable components using a set-level objective that directly reflects the final Best-of-N selection criterion. Let

$$\mathcal{I}^\star(x, \boldsymbol{y}(x)) = \left\{ i \in [N] \;\middle|\; r^\star(x, y_i) = \max_{j \in [N]} r^\star(x, y_j) \right\} \tag{8}$$

denote the indices of candidates that attain the highest ground-truth reward within the current collection. This definition allows multiple candidates to be optimal when their ground-truth rewards are tied. The calibrated scores induce a probability distribution over the candidate responses:

$$p_\theta(i \mid x, \boldsymbol{y}(x)) = \frac{\exp\left(\tilde{r}_\theta(x, y_i)\right)}{\sum_{j=1}^{N} \exp\left(\tilde{r}_\theta(x, y_j)\right)}. \tag{9}$$

We optimize the following set-level selection loss:

$$\mathcal{L}_\theta(x, \boldsymbol{y}(x)) := -\log \sum_{i \in \mathcal{I}^\star(x)} p_\theta(i \mid x, \boldsymbol{y}(x)) \tag{10}$$

The loss encourages LESS to assign high probability mass to candidates with maximal ground-truth reward. When there is a unique optimal candidate, it reduces to the standard cross-entropy loss over candidate indices. When multiple candidates are optimal, their probabilities are aggregated so that selecting any optimal response is treated as correct.

In practice, let the training set be $\mathcal{D} = \left\{ \left(x_k, \boldsymbol{y}^{(N_{\max})}(x_k), \boldsymbol{r}^{\star,(N_{\max})}(x_k)\right) \right\}_{k=1}^{K}$, where $K = |\mathcal{D}|$ is the number of training prompts, $\boldsymbol{y}^{(N_{\max})}(x_k)$ is the pre-generated candidate pool containing $N_{\max}$ responses associated with prompt x_k, and $\boldsymbol{r}^{\star,(N_{\max})}(x_k)$ contains their corresponding ground-truth rewards. During training, for each prompt x, we sample a candidate-set size N from $\mathcal{N} := \{2, 3, \ldots, N_{\max}\}$ and randomly subsample N responses without replacement from $\boldsymbol{y}^{(N_{\max})}(x)$ to construct $\boldsymbol{y}(x)$. The resulting prompt–candidate-set pair is treated as one set-level training example. By exposing the selector to candidate sets of varying sizes, this construction enables a single model to generalize across different inference-time sampling budgets. The algorithmic description of LESS is presented in Algorithm 1.

Algorithm 1 LESS Algorithm

Require: Set-wise selector f_θ, proxy reward model $\hat{r}$, embedding model E_{emb}, training set $\mathcal{D} = \left\{\left(x_k, \boldsymbol{y}^{(N_{\max})}(x_k), \boldsymbol{r}^{\star,(N_{\max})}(x_k)\right)\right\}_{k=1}^{K}$, candidate-set size set $\mathcal{N} = \{2, 3, \ldots, N_{\max}\}$, and mini-batch size B.

1: Initialize the set-wise selector f_θ.
2: **while** not converged **do**
3: Sample a mini-batch of training prompts $\mathcal{B} = \{x_b\}_{b=1}^{B}$ from $\mathcal{D}$.
4: **for** $x_b \in \mathcal{B}$ **do**
5: Sample a candidate-set size N from $\mathcal{N}$ and construct $\boldsymbol{y}(x_b) = [y_{b,1}, \ldots, y_{b,N}]$ by randomly subsampling N responses from $\boldsymbol{y}^{(N_{\max})}(x_b)$.
6: For each candidate $y_{b,j}$, construct its reward-statistic feature $\boldsymbol{a}_{b,j}$ and prompt–response embedding $\boldsymbol{e}_{b,j}$ using Eq. (3) and Eq. (4).
7: Obtain contextualized candidate representations $\mathbf{C}(x_b, \boldsymbol{y}(x_b))$ using Eq. (5) and Eq. (6).
8: Get corresponding calibrated selection scores $\tilde{\boldsymbol{r}}(x_b, \boldsymbol{y}(x_b))$ using Eq. (7).
9: **end for**
10: Compute the batch of set-wise selection losses $\{\mathcal{L}_\theta(x_b, \boldsymbol{y}(x_b))\}_{b=1}^{B}$ using Eq. (10) and update θ with the averaged loss.
11: **end while**

Ensure: Learned set-wise selector f_θ.

3.4 Theoretical Analysis

To quantify the inference-time performance of LESS for a given prompt x, we consider the following per-prompt regret:

$$\mathrm{Reg}_{\theta,N}(x) := \max_{y \in \mathcal{Y}} r^\star(x, y) - \mathbb{E}_{\boldsymbol{y}(x)}\left[r^\star(x, y_{i_\theta(x, \boldsymbol{y}(x))})\right]. \tag{11}$$

This regret measures the gap between the globally optimal ground-truth reward and the expected ground-truth reward of the response selected by LESS. The following proposition upper bounds this regret by separating the effect of candidate coverage from the error introduced by the learned set-wise selector.

Proposition 1 *For a prompt $x \in \mathcal{X}$ and a candidate-set size $N \geq 2$, suppose that $r^\star(x, y) \in [0, r_{\max}]$ for every $y \in \mathcal{Y}$ and that $\max_{y \in \mathcal{Y}} r^\star(x, y)$ exists. Let $m^\star(x, \boldsymbol{y}(x)) := |\mathcal{I}^\star(x, y(x))|$ denote the number of candidates attaining the highest ground-truth reward in $\boldsymbol{y}(x)$, and let $\Delta_N^\star(x, \boldsymbol{y}(x)) := \max_{j \in [N]} r^\star(x, y_j) - \min_{j \in [N]} r^\star(x, y_j)$ denote the ground-truth reward range within the candidate set. For each $\epsilon \in [0, r_{\max}]$, let $q_\epsilon(x) := \Pr_{y \sim \pi(\cdot|x)}(r^\star(x, y) \geq \max_{y' \in \mathcal{Y}} r^\star(x, y') - \epsilon)$ denote the probability that a single response sampled from the base policy attains a ground-truth reward within ϵ of the globally optimal reward. Then*

$$\mathrm{Reg}_{\theta,N}(x) \leq \int_0^{r_{\max}} \left(1 - q_\epsilon(x)\right)^N d\epsilon + \mathbb{E}_{\boldsymbol{y}(x)}\left[\frac{\Delta_N^\star(x, \boldsymbol{y}(x)) \mathcal{L}_\theta(x, \boldsymbol{y}(x))}{\log(1 + 1/m^\star(x, \boldsymbol{y}(x))}\right]. \tag{12}$$

The proof of Proposition 1 is provided in Appendix A. The first term in Eq. (12) characterizes the expected candidate-coverage gap. As N increases, the probability that all sampled candidates have ground-truth rewards more than ϵ below the globally optimal reward decreases, making this term monotonically non-increasing in N. The second term upper-bounds the error introduced by the set-wise selection rule in terms of the set-wise loss, the within-set ground-truth reward range, and the number of optimal candidates. In particular, for a fixed candidate set, assigning greater total probability mass to the ground-truth-optimal candidates decreases the set-wise loss and hence tightens the resulting upper bound on the selection error.

4 Experiments

4.1 Datasets

We evaluate LESS on the `Preference Proxy Evaluations` (`PPE`) benchmark (Frick et al., 2025). `PPE` includes five subsets for Best-of-N evaluation: `MMLU-Pro` (Wang et al., 2024), `MATH`

(Hendrycks et al., 2021), GPQA (Rein et al., 2024), MBPP-Plus (Austin et al., 2021), and IFEval (Zhou et al., 2023). These subsets cover general knowledge, mathematics, STEM, coding, and instruction following, respectively. Each subset contains approximately 512 prompts. Each prompt is associated with 32 independently sampled responses from one of four generation models, namely Llama-3-8B-Instruct (Grattafiori et al., 2024), Gemma-2-9B-it (Gemma Team et al., 2024), Claude-3-Haiku, and GPT-4o-mini. Each prompt is accompanied by verifiable correctness labels, enabling evaluation across different values of N. Complete dataset details are given in Appendix B.

4.2 BASELINES

We compare LESS with the following representative Best-of-N alignment baselines:

- BON (Brown et al., 2024): A standard Best-of-N approach that samples N candidate responses from the reference LLM and selects the one with the highest reward-model score.
- MBR-BON (Jinnai et al., 2025): A regularized Best-of-N approach that combines reward maximization with an MBR-based proximity term, preferring candidates that are both highly rewarded and representative of the sampled response distribution.
- SOFT BON (Verdun et al., 2025): A temperature-controlled Best-of-N approach that replaces hard top-reward selection with temperature-controlled probabilistic selection.
- INFERENCETIMEPESSIMISM (ITP) (Huang et al., 2025): A pessimistic Best-of-N approach that uses χ^2-regularization to downweight uncertain high-reward candidates and prevents reward overoptimization as the sampling budget increases.
- BEST-OF-TAILS (Hsu et al., 2026): A tail-adaptive Best-of-N approach that estimates reward tail behavior and dynamically interpolates between optimistic and pessimistic selection strategies.
- SELF CONSISTENCY (Wang et al., 2023): An inference-time sampling baseline that generates multiple candidate reasoning paths and selects the final answer by majority voting. We apply it to the multiple-choice benchmarks MMLU-Pro and GPQA.

4.3 IMPLEMENTATION DETAILS

We use Skywork-Reward-V2-Qwen3-1.7B, Skywork-Reward-V2-Llama-3.2-3B (Liu et al., 2025a) and Ai2 Llama-3.1-Tulu-3-8B-RM (Lambert et al., 2024) as the proxy reward models. For each candidate set, we normalize the proxy-reward feature and the reward-gap feature before feeding them into the selector. For both LESS and the MBR-BON, we obtain prompt-response embeddings using Qwen3-Embedding-0.6B (Zhang et al., 2025b). The embedding dimension is $d_e = 1024$. We set $d_r = 16$ and $d_s = 48$, resulting in a $d = 64$-dimensional candidate representation. The encoder g_{set} consists of two stacked SABs. The maximum input length for the embedding model and all reward models is 4096 tokens.

For each subset of PPE, we randomly select 200 prompts for training and use the remaining prompts for evaluation. For LESS, the training split is used to optimize the selector network. For the competing baselines, the same training split is used to tune their hyperparameters following the procedures described in their original papers. To improve the selector's ability to generalize across different values of N, we construct training candidate sets with varying sizes by randomly subsampling from the full candidate set. We train the selector using the AdamW optimizer (Loshchilov & Hutter, 2019) for 2 epochs with a learning rate of 1×10^{-4}, a weight decay of 1×10^{-4}, and a batch size of 64.

4.4 EXPERIMENTAL RESULTS

For evaluation, we randomly subsample each prompt's full candidate set to simulate the varying candidate-set sizes encountered in practical Best-of-N inference. We evaluate values of N at intervals of 2. For each value of N, we repeat the random subsampling 5 times and report the mean accuracy together with the standard deviation. The results are presented in Figure 1, with additional results provided in Figure 6 in the appendix. The results show that:

- LESS consistently improves over standard Best-of-N across five benchmarks covering different domains and three reward models, and outperforms the competing baselines in most settings. These results demonstrate the broad applicability and effectiveness of LESS under varying inference-time conditions.

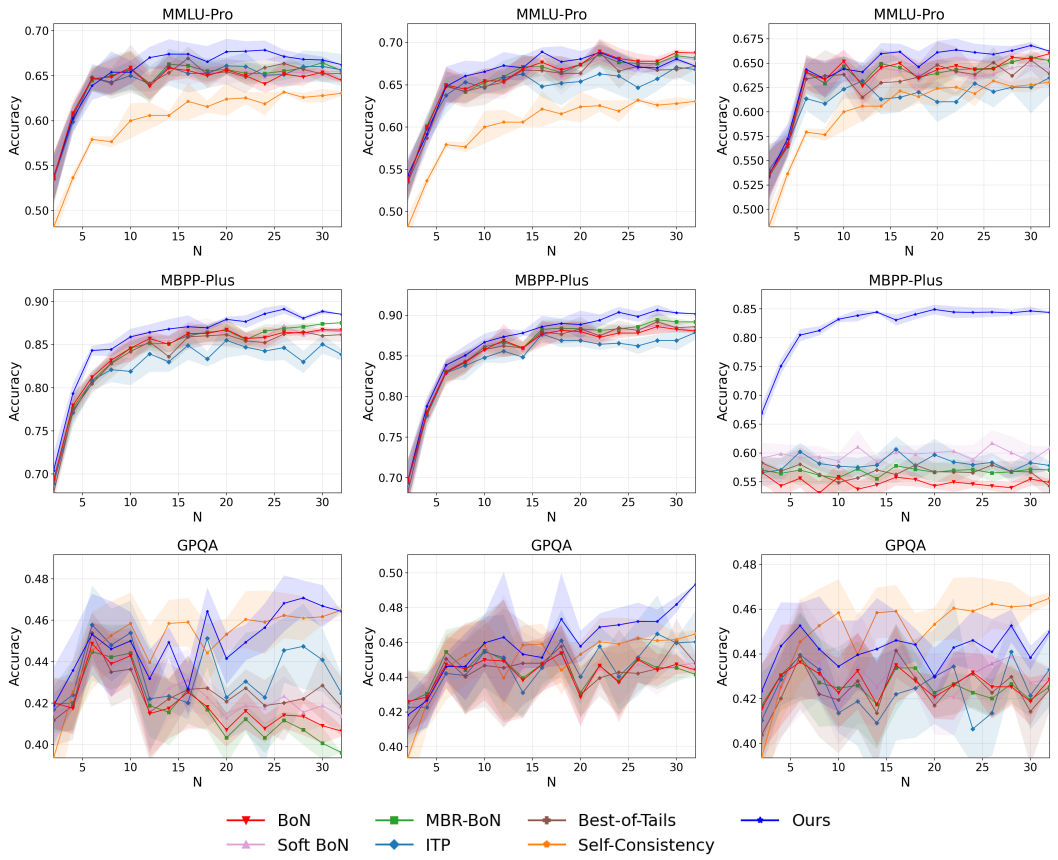


Figure 1: Best-of-N accuracy of LESS and the compared methods on `MMLU-Pro`, `MBPP-Plus`, and `GPQA` across different values of N. The first, second, and third columns use Skywork-Reward-V2-Qwen3-1.7B, Skywork-Reward-V2-Llama-3.2-3B, and Ai2 Llama-3.1-Tulu-3-8B-RM as the reward model, respectively.

- As N increases, the Best-of-N accuracy of LESS generally exhibits an upward trend, indicating that our method can effectively leverage additional test-time compute rather than suffering from performance degradation when more candidates are considered.
- The improvements of LESS are particularly pronounced in settings where reward hacking is more evident, such as those observed on the `GPQA` dataset. This suggests that the learned set-wise selector can identify effective selection strategies under challenging inference-time conditions where proxy rewards are less reliable.
- For the combination of `MBPP-Plus` and Ai2 Llama-3.1-Tulu-3-8B-RM, the reward model appears to have limited ability to distinguish higher-quality coding responses, resulting in relatively poor raw Best-of-N performance. Under this challenging setting, all calibration baselines substantially improve Best-of-N accuracy, while LESS outperforms them by a substantial margin.

4.5 Ablation Study

We conduct ablation study with three variants of LESS: LESS-NE, LESS-NR, and LESS-NS. In LESS-NE, we remove the prompt-response embeddings and retain only the reward-statistic features. In LESS-NR, we remove all reward-related features, including the proxy reward and the derived reward statistics, and rely solely on the semantic embeddings. In LESS-NS, we replace the set-wise encoder with a candidate-wise MLP of comparable parameter count. The MLP processes each candidate representation independently without exchanging information across candidates, allowing us to isolate the contribution of set-wise contextualization while controlling for model capacity. We conduct experiments on `MMLU-Pro`, `MBPP-Plus`, and `GPQA` across different values

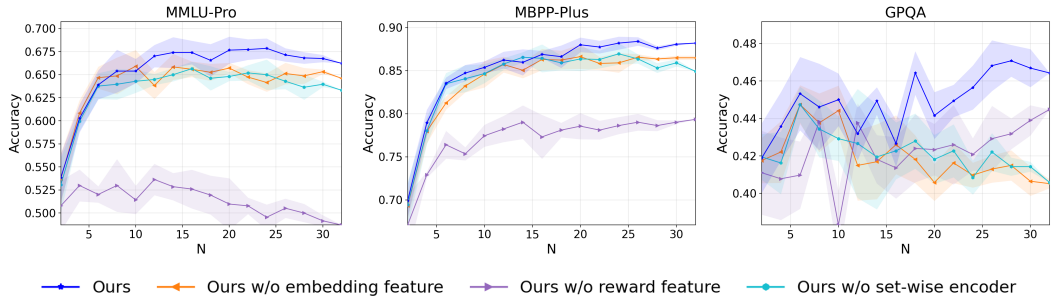
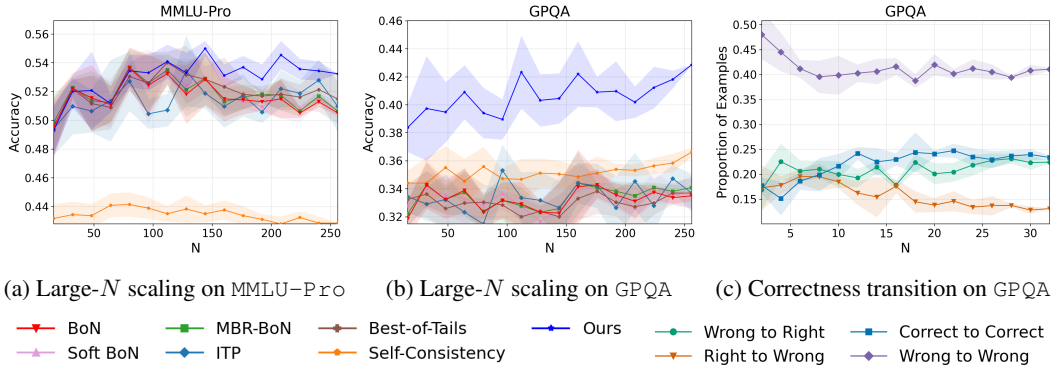


Figure 2: Best-of-N accuracy of LESS and its variants on `MMLU-Pro`, `MBPP-Plus`, and `GPQA` across different values of N with Skywork-Reward-V2-Qwen3-1.7B used as the reward model.



(a) Large-N scaling on `MMLU-Pro` (b) Large-N scaling on `GPQA` (c) Correctness transition on `GPQA`

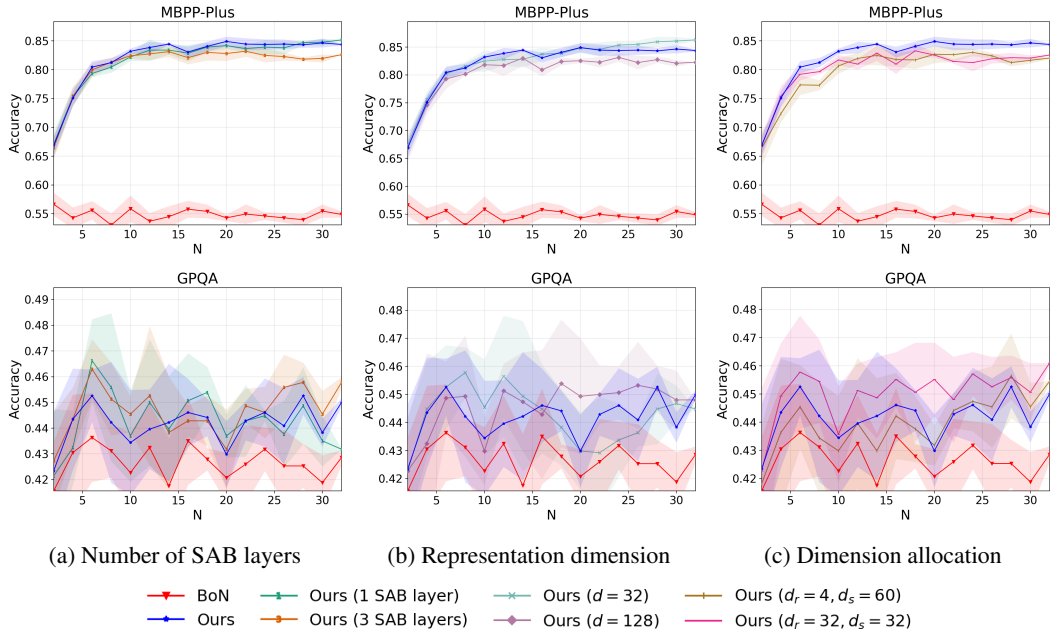
Figure 3: Best-of-N accuracy of LESS and the compared methods on `MMLU-Pro` and `GPQA` in the large-N regime, together with the correctness transition proportion from standard Best-of-N to LESS on `GPQA`, using Skywork-Reward-V2-Qwen3-1.7B as the reward model.

of N with Skywork-Reward-V2-Qwen3-1.7B used as the reward model. The results in Figure 2 show that both reward features and embedding features make important contributions to the final selection accuracy. In particular, the embedding-only variant suffers a more substantial performance degradation, suggesting that semantic representations alone are insufficient to reliably identify the best candidate. Removing the set-wise encoder also consistently reduces accuracy, demonstrating the benefit of contextualizing each response with respect to the other $N-1$ candidates and capturing their pairwise and higher-order relationships during selection.

4.6 Further Analysis

Scaling to Large Candidate Sets. To examine the behavior of LESS under extremely large candidate-set sizes, we resample 256 responses for each prompt in the `MMLU-Pro` and `GPQA` subsets of `PPE` using Llama-3-8B-Instruct, and train the selector on 200 prompts from each dataset. Figures 3a and 3b present the results using Skywork-Reward-V2-Qwen3-1.7B as the reward model, while Figure 7 reports additional results with Skywork-Reward-V2-Llama-3.2-3B and Ai2 Llama-3.1-Tulu-3-8B-RM. Across these settings, LESS maintains strong performance as the candidate-set size increases to $N = 256$. Together with the results in the smaller-N regime, these results demonstrate that LESS remains effective across a wide range of inference-time sampling budgets.

Analysis of Selection Revisions. To better understand the selection behavior of LESS, we analyze both how frequently it revises the decisions made by standard Best-of-N and the outcomes of these revisions. Figure 8 reports the selection revision rate, i.e., the proportion of examples for which LESS selects a response different from the one chosen by Best-of-N. We observe that the revision rate generally increases with N, indicating that LESS intervenes more frequently as the candidate set becomes larger. To further examine whether these revisions are beneficial, Figures 3c and 9 decompose the resulting correctness transitions into different categories. In particular, beneficial revi-



(a) Number of SAB layers (b) Representation dimension (c) Dimension allocation

Figure 4: Sensitivity analysis of LESS with respect to the SAB encoder depth, candidate representation dimension, and reward-semantic dimension allocation on `MBPP-Plus` and `GPQA`, using Ai2 Llama-3.1-Tulu-3-8B-RM as the reward model.

sions, where LESS changes an incorrect Best-of-N selection into a correct one (wrong-to-correct), occur more frequently overall than harmful revisions that change a correct selection into an incorrect one (correct-to-wrong).

Sensitivity to Architectural Choices. We conduct a sensitivity analysis on the depth and hidden dimension of the SAB-based set encoder. For the encoder depth, we vary the number of SAB layers over $\{1, 2, 3\}$, while for the hidden dimension, we consider $d \in \{32, 64, 128\}$. The results are shown in Figure 4a and 4b, respectively. Although some configurations achieve slightly better performance than our default setting, the overall performance remains stable across different architectural choices, and all tested configurations consistently outperform standard Best-of-N. These results suggest that the effectiveness of LESS is not sensitive to a particular choice of encoder depth or hidden dimension. Besides, we conduct a sensitivity analysis on the allocation of representation dimensions between the reward-statistic and semantic features. Keeping the total representation dimension fixed at $d = 64$, we consider three settings, with $d_r = 16$ and $d_s = 48$ as the default, and $d_r = 4$, $d_s = 60$ and $d_r = 32$, $d_s = 32$ as alternatives. The results are shown in Figure 4c. The overall performance remains stable across different dimension allocations, indicating that the effectiveness of LESS is not sensitive to a particular allocation between the two feature types. More analysis can be found in Appendix C and some case studies can be found in Appendix D.

5 Conclusion

In this paper, we propose LESS, a learned set-wise selection framework for Best-of-N alignment. Rather than relying on fixed rules, LESS learns a tailored selection strategy directly from the corresponding data to select the best response among the N candidates. It represents each candidate response using proxy-reward features and the prompt-response embedding, then employs a set transformer encoder to capture interactions among N candidates. The model assigns a selection logit to each response and is trained with a set-level objective tailored to final response selection, with the highest-scoring candidate chosen as the output. The set-based design naturally accommodates varying candidate-set sizes, providing a scalable and data-driven approach as inference-time compute increases. Extensive experiments across diverse benchmarks and varying values of N demonstrate the effectiveness of the proposed approach.

AI USE STATEMENT

In this work, we have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results. Generating synthetic datasets and cleaning or reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to create or edit software code and to edit the research paper to improve readability. We reviewed all AI-assisted work, including checking AI-assisted code for correctness and manually reviewing all AI-edited text to ensure that it accurately reflects the authors' intended meaning and the technical content of the paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our methodology and training procedure in Section 3 and Algorithm 1. The datasets and data splits used in our experiments are described in Section 4.1 and Appendix B. Section 4.3 provides the implementation and evaluation details. Finally, the complete proof of our theoretical result is provided in Appendix A. Together, these details are intended to support reproduction of both the training procedure and the reported experimental results.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint*, arXiv:2108.07732, 2021.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint*, arXiv:2204.05862, 2022.

Andre Barreto, Vincent Dumoulin, Yiran Mao, Mark Rowland, Nicolas Perez-Nieves, Bobak Shahriari, Yann Dauphin, Doina Precup, and Hugo Larochelle. Capturing individual human preferences with reward features. In *Advances in Neural Information Processing Systems 38*, pp. 82119–82154, San Diego, CA, 2025.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 3580–3602, Vancouver, Canada, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint*, arXiv:2407.21787, 2024.

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.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria, 2024.

Zhibin Duan, Guowei Rong, Zhuo Li, Bo Chen, Mingyuan Zhou, and Dandan Guo. Mitigating reward hacking in rlhf via bayesian non-negative reward modeling. In *Proceedings of the 43rd International Conference on Machine Learning*, pp. in press, Seoul, South Korea, 2026.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alexander Nicholas D'Amour, Krishnamurthy Dj Dvijotham, Adam Fisch, Katherine A Heller, Stephen Robert Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *Proceedings of the 1st Conference on Language Modeling*, Philadelphia, PA, 2024.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. How to evaluate reward models for rlhf. In *Proceedings of the 13th International Conference on Learning Representations*, pp. 18128–18163, Singapore, 2025.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10835–10866, Honolulu, HI, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint*, arXiv:2408.00118, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, et al. The llama 3 herd of models. *arXiv preprint*, arXiv:2407.21783, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks*, Virtual Event, 2021.

Jiwoo Hong, Noah Lee, Eunki Kim, Guijin Son, Woojin Chung, Aman Gupta, Shao Tang, and James Thorne. On the robustness of reward models for language model alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 23682–23699, Vancouver, Canada, 2025.

Hsiang Hsu, Eric Lei, and Chun-Fu Chen. Best-of-tails: Bridging optimism and pessimism in inference-time alignment. *arXiv preprint*, arXiv:2603.06797, 2026.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*, Virtual Event, 2022.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 25075–25126, Vancouver, Canada, 2025.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kenshi Abe, Kaito Ariu, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of best-of-n sampling strategies for language model alignment. *Transactions on Machine Learning Research*, 2025.

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling with minimum bayes risk objective for language model alignment. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9321–9347, Albuquerque, New Mexico, 2025.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio Calmon. Inference-time reward hacking in large language models. In *Advances in Neural Information Processing Systems 38*, pp. 61720–61760, San Diego, CA, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint*, arXiv:2411.15124, 2024.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3744–3753, Long Beach, CA, 2019.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Zhuo Li, Yuege Feng, Dandan Guo, Jinpeng Hu, Anningzhe Gao, and Xiang Wan. Aplot: Robust reward modeling via adaptive preference learning with optimal transport. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5524–5538, Suzhou, China, 2025.

Zhuo Li, Pengyu Cheng, Zhechao Yu, Feifei Tong, Anningzhe Gao, Tsung-Hui Chang, Xiang Wan, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Eliminating inductive bias in reward models with information-theoretic guidance. In *Proceedings of the 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026.

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 *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria, 2024.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint*, arXiv:2410.18451, 2024.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint*, arXiv:2507.01352, 2025a.

Tianqi Liu, Wei Xiong, Jie Ren, Lichang Chen, Junru Wu, Rishabh Joshi, Yang Gao, Jiaming Shen, Zhen Qin, Tianhe Yu, Daniel Sohn, Anastasia Makarova, Jeremiah Zhe Liu, Yuan Liu, Bilal Piot, Abe Ittycheriah, Aviral Kumar, and Mohammad Saleh. Rrm: Robust reward model training mitigates reward hacking. In *Proceedings of the 13th International Conference on Learning Representations*, pp. 62682–62700, Singapore, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA, 2019.

Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling. In *Advances in Neural Information Processing Systems 37*, pp. 134387–134429, Vancouver, Canada, 2024.

Birong Pan, Yongqi Li, Weiyu Zhang, Wenpeng Lu, Mayi Xu, Shen Zhou, Yuanyuan Zhu, Ming Zhong, and Tieyun Qian. A survey on training-free alignment of large language models. *arXiv preprint*, arXiv:2508.09016, 2025.

Alexandre Rame, Nino Vieillard, Leonard Hussenot, Robert Dadashi-Tazehozi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 42048–42073, Vienna, Austria, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *Proceedings of the 1st Conference on Language Modeling*, Philadelphia, PA, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems 33*, pp. 3008–3021, Virtual Event, 2020.

Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio P. Calmon. Soft best-of-n sampling for model alignment. In *Proceedings of the 2025 IEEE International Symposium on Information Theory*, Ann Arbor, MI, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning. In *Advances in Neural Information Processing Systems 38*, pp. 143630–143664, San Diego, CA, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37*, pp. 95266–95290, Vancouver, Canada, 2024.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. In *Advances in Neural Information Processing Systems 37*, pp. 62279–62309, Vancouver, Canada, 2024.

Zhuohao Yu, Steven Wu, and Adam Block. From curiosity to caution: Mitigating reward hacking for best-of-n with pessimism. In *Proceedings of the 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. In *Advances in Neural Information Processing Systems 30*, pp. 3391–3401, Long Beach, CA, 2017.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint*, arXiv:2503.24235, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint*, arXiv:2506.05176, 2025b.

Zhaowei Zhang, Fengshuo Bai, Qizhi Chen, Chengdong Ma, Mingzhi Wang, Haoran Sun, Zilong Zheng, and Yaodong Yang. Amulet: Realignment during test time for personalized preference adaptation of llms. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore, 2025c.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint*, arXiv:2311.07911, 2023.

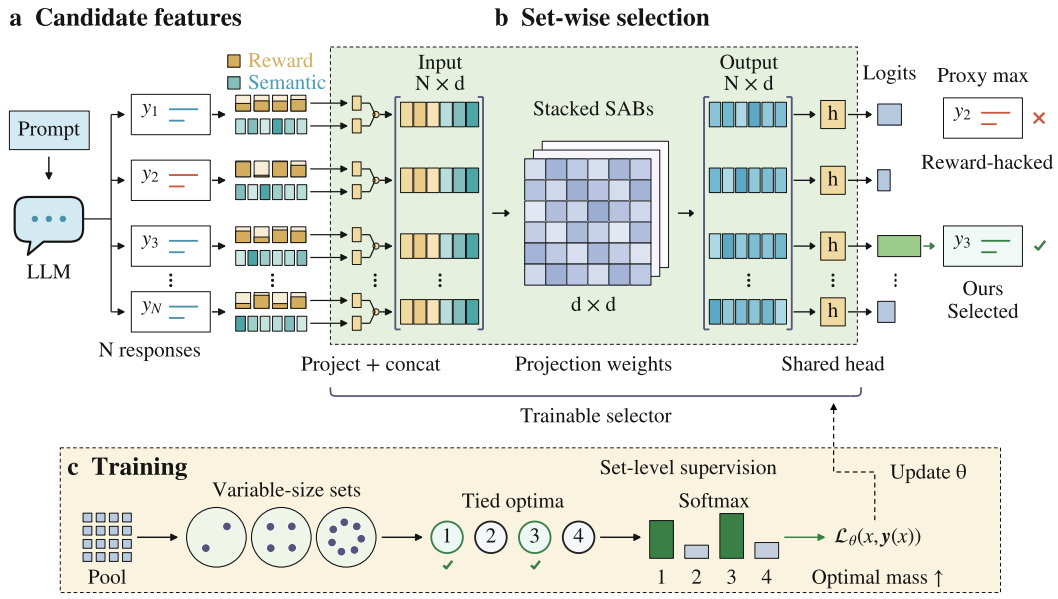


Figure 5: The LESS framework.

A Proof of Proposition 1

Proof. Fix $x \in \mathcal{X}$ and write $\boldsymbol{y} = \boldsymbol{y}(x)$ for brevity. We first decompose the regret by adding and subtracting the expected highest ground-truth reward in the candidate set:

$$\begin{aligned}
\mathrm{Reg}_{\theta,N}(x) &= \max_{y\in\mathcal{Y}} r^\star(x,y) - \mathbb{E}\left[r^\star(x, y_{i_\theta(x,\boldsymbol{y})})\right] \\
&= \max_{y\in\mathcal{Y}} r^\star(x,y) - \mathbb{E}\left[\max_{j\in[N]} r^\star(x,y_j)\right] + \mathbb{E}\left[\max_{j\in[N]} r^\star(x,y_j)\right] - \mathbb{E}\left[r^\star(x, y_{i_\theta(x,\boldsymbol{y})})\right] \\
&= \underbrace{\max_{y\in\mathcal{Y}} r^\star(x,y) - \mathbb{E}\left[\max_{j\in[N]} r^\star(x,y_j)\right]}_{\text{candidate-coverage term}} + \underbrace{\mathbb{E}\left[\max_{j\in[N]} r^\star(x,y_j) - r^\star(x, y_{i_\theta(x,\boldsymbol{y})})\right]}_{\text{set-wise selection-error term}}.
\end{aligned} \tag{13}$$

Then, we bound the two terms in Eq. (13) separately. For the candidate-coverage term, define the nonnegative random variable

$$Z_N := \max_{y\in\mathcal{Y}} r^\star(x,y) - \max_{j\in[N]} r^\star(x,y_j).$$

Because $r^\star(x,y) \in [0, r_{\max}]$, we have $0 \le Z_N \le r_{\max}$. By the tail-integral identity for nonnegative random variables,

$$\mathbb{E}[Z_N] = \int_0^{r_{\max}} \Pr(Z_N > \epsilon)\, d\epsilon. \tag{14}$$

For every $\epsilon \in [0, r_{\max}]$, the event $Z_N > \epsilon$ can be written as

$$\begin{aligned}
\{Z_N > \epsilon\} &= \left\{\max_{j\in[N]} r^\star(x,y_j) < \max_{y\in\mathcal{Y}} r^\star(x,y) - \epsilon\right\} \\
&= \bigcap_{j=1}^{N} \left\{r^\star(x,y_j) < \max_{y\in\mathcal{Y}} r^\star(x,y) - \epsilon\right\}.
\end{aligned} \tag{15}$$

The second equality holds because the maximum sampled reward is below the threshold if and only if every sampled reward is below the threshold. Since $y_1, \ldots, y_N$ are sampled independently and

identically from $\pi(\cdot \mid x)$,

$$\begin{aligned}\Pr(Z_N > \epsilon) &= \prod_{j=1}^{N} \Pr_{y_j \sim \pi(\cdot|x)} \left(r^\star(x, y_j) < \max_{y \in \mathcal{Y}} r^\star(x, y) - \epsilon \right) \\ &= \left[1 - \Pr_{y \sim \pi(\cdot|x)} \left(r^\star(x, y) \geq \max_{y' \in \mathcal{Y}} r^\star(x, y') - \epsilon \right) \right]^N \\ &= \big(1 - q_\epsilon(x)\big)^N.\end{aligned} \tag{16}$$

Substituting Eq. (16) into Eq. (14) and using the definition of Z_N gives

$$\max_{y \in \mathcal{Y}} r^\star(x, y) - \mathbb{E}\left[\max_{j \in [N]} r^\star(x, y_j) \right] = \int_0^{r_{\max}} \big(1 - q_\epsilon(x)\big)^N \, d\epsilon. \tag{17}$$

We next control the set-wise selection-error term. For a fixed candidate set, define the total probability assigned to the ground-truth-optimal candidates as

$$Q_\theta(x, \boldsymbol{y}) := \sum_{i \in \mathcal{I}^\star(x, \boldsymbol{y})} p_\theta(i \mid x, \boldsymbol{y}).$$

By the definition of the set-wise loss, $\mathcal{L}_\theta(x, \boldsymbol{y}) = -\log Q_\theta(x, \boldsymbol{y})$. Since $p_\theta(\cdot \mid x, \boldsymbol{y})$ is induced by a softmax, $Q_\theta(x, \boldsymbol{y}) \in (0, 1]$ and hence $\mathcal{L}_\theta(x, \boldsymbol{y}) \geq 0$. Moreover, $\mathcal{I}^\star(x, \boldsymbol{y})$ is nonempty, so $m^\star(x, \boldsymbol{y}) \geq 1$ and $\log(1 + 1/m^\star(x, \boldsymbol{y})) > 0$.

We first prove the following pointwise inequality:

$$\mathbf{1}\left\{ i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y}) \right\} \leq \frac{\mathcal{L}_\theta(x, \boldsymbol{y})}{\log(1 + 1/m^\star(x, \boldsymbol{y}))}. \tag{18}$$

Suppose first that $i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y})$, and denote the probability assigned to the selected candidate by

$$p_{\text{sel}} := p_\theta(i_\theta(x, \boldsymbol{y}) \mid x, \boldsymbol{y}).$$

Since $i_\theta(x, \boldsymbol{y})$ maximizes the calibrated score and softmax preserves the ordering of its inputs, $i_\theta(x, \boldsymbol{y})$ also maximizes $p_\theta(i \mid x, \boldsymbol{y})$. Therefore,

$$p_\theta(i \mid x, \boldsymbol{y}) \leq p_{\text{sel}} \qquad \text{for every } i \in \mathcal{I}^\star(x, \boldsymbol{y}).$$

Summing this inequality over all $m^\star(x, \boldsymbol{y})$ elements of $\mathcal{I}^\star(x, \boldsymbol{y})$ yields

$$Q_\theta(x, \boldsymbol{y}) = \sum_{i \in \mathcal{I}^\star(x, \boldsymbol{y})} p_\theta(i \mid x, \boldsymbol{y}) \leq m^\star(x, \boldsymbol{y}) p_{\text{sel}}. \tag{19}$$

Because $i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y})$, the probability p_{sel} is included in the total probability mass assigned outside the optimal-index set. Hence

$$p_{\text{sel}} \leq \sum_{i \notin \mathcal{I}^\star(x, \boldsymbol{y})} p_\theta(i \mid x, \boldsymbol{y}) = 1 - Q_\theta(x, \boldsymbol{y}). \tag{20}$$

Combining Eq. (19) and Eq. (20) gives

$$Q_\theta(x, \boldsymbol{y}) \leq m^\star(x, \boldsymbol{y}) \big(1 - Q_\theta(x, \boldsymbol{y})\big).$$

Rearranging this inequality,

$$Q_\theta(x, \boldsymbol{y}) + m^\star(x, \boldsymbol{y}) Q_\theta(x, \boldsymbol{y}) \leq m^\star(x, \boldsymbol{y}),$$

so

$$\big(m^\star(x, \boldsymbol{y}) + 1\big) Q_\theta(x, \boldsymbol{y}) \leq m^\star(x, \boldsymbol{y}).$$

Because $m^\star(x, \boldsymbol{y}) + 1 > 0$, it follows that

$$Q_\theta(x, \boldsymbol{y}) \leq \frac{m^\star(x, \boldsymbol{y})}{m^\star(x, \boldsymbol{y}) + 1}. \tag{21}$$

The function $z \mapsto -\log z$ is decreasing on $(0, 1]$. Applying it to Eq. (21) therefore gives

$$\begin{aligned}\mathcal{L}_\theta(x, \boldsymbol{y}) &= -\log Q_\theta(x, \boldsymbol{y}) \\ &\geq -\log\left(\frac{m^\star(x, \boldsymbol{y})}{m^\star(x, \boldsymbol{y}) + 1}\right) = \log\left(\frac{m^\star(x, \boldsymbol{y}) + 1}{m^\star(x, \boldsymbol{y})}\right) = \log\left(1 + \frac{1}{m^\star(x, \boldsymbol{y})}\right). \end{aligned} \tag{22}$$

Consequently, on the event $i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y})$,

$$\mathbf{1}\left\{i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y})\right\} = 1 \leq \frac{\mathcal{L}_\theta(x, \boldsymbol{y})}{\log(1 + 1/m^\star(x, \boldsymbol{y}))}.$$

On the complementary event $i_\theta(x, \boldsymbol{y}) \in \mathcal{I}^\star(x, \boldsymbol{y})$, the indicator is zero, whereas the right-hand side is nonnegative. Thus, Eq. (18) holds for every candidate set.

We now translate this indicator bound into a ground-truth reward bound. If $i_\theta(x, \boldsymbol{y}) \in \mathcal{I}^\star(x, \boldsymbol{y})$, then

$$\max_{j \in [N]} r^\star(x, y_j) - r^\star(x, y_{i_\theta(x, \boldsymbol{y})}) = 0.$$

If $i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y})$, then $r^\star(x, y_{i_\theta(x, \boldsymbol{y})}) \geq \min_{j \in [N]} r^\star(x, y_j)$, and therefore

$$\max_{j \in [N]} r^\star(x, y_j) - r^\star(x, y_{i_\theta(x, \boldsymbol{y})}) \leq \Delta_N^\star(x, \boldsymbol{y}).$$

Combining the two cases yields the pointwise inequality

$$\max_{j \in [N]} r^\star(x, y_j) - r^\star(x, y_{i_\theta(x, \boldsymbol{y})}) \leq \Delta_N^\star(x, \boldsymbol{y})\, \mathbf{1}\left\{i_\theta(x, \boldsymbol{y}) \notin \mathcal{I}^\star(x, \boldsymbol{y})\right\}. \tag{23}$$

Applying Eq. (18) to Eq. (23) gives

$$\max_{j \in [N]} r^\star(x, y_j) - r^\star(x, y_{i_\theta(x, \boldsymbol{y})}) \leq \frac{\Delta_N^\star(x, \boldsymbol{y}) \mathcal{L}_\theta(x, \boldsymbol{y})}{\log(1 + 1/m^\star(x, \boldsymbol{y}))}.$$

Taking expectation over the random candidate set gives

$$\mathbb{E}\left[\max_{j \in [N]} r^\star(x, y_j) - r^\star(x, y_{i_\theta(x, \boldsymbol{y})})\right] \leq \mathbb{E}\left[\frac{\Delta_N^\star(x, \boldsymbol{y}) \mathcal{L}_\theta(x, \boldsymbol{y})}{\log(1 + 1/m^\star(x, \boldsymbol{y}))}\right]. \tag{24}$$

Finally, substituting Eq. (17) and Eq. (24) into the decomposition in Eq. (13) proves Eq. (12).

B Dataset Details

`PPE` includes five verifiable benchmarks spanning diverse capabilities, including general knowledge, mathematical reasoning, scientific reasoning, code generation, and instruction following.

- `MMLU-Pro` (Wang et al., 2024) is a challenging multi-domain benchmark designed to evaluate both the knowledge and reasoning abilities of language models. It contains more than 12,000 multiple-choice questions spanning 14 domains, including mathematics, physics, chemistry, law, engineering, psychology, and health, with each question providing ten answer choices.
- `MATH` (Hendrycks et al., 2021) is a mathematical reasoning benchmark consisting of 12,500 challenging problems collected from high-school mathematics competitions. The problems span seven mathematical subjects and multiple difficulty levels, and each problem is accompanied by a step-by-step solution and a final answer.
- `GPQA` (Rein et al., 2024) is a highly challenging multiple-choice benchmark containing questions written by domain experts in biology, physics, and chemistry. The questions require graduate-level domain knowledge and are difficult even for skilled non-experts with unrestricted access to external resources.
- `MBPP-Plus` (Austin et al., 2021) is a code-generation benchmark built upon the Mostly Basic Programming Problems (`MBPP`) dataset, which evaluates the ability of language models to synthesize short Python programs from natural-language specifications. The generated programs are evaluated through executable test cases, providing an objective measure of functional correctness.
- `IFEval` (Zhou et al., 2023) evaluates the instruction-following ability of language models using prompts containing objectively verifiable constraints. It defines 25 types of verifiable instructions, covering requirements on output length, keywords, formatting, and response structure, and contains 541 prompts with one or more such constraints.

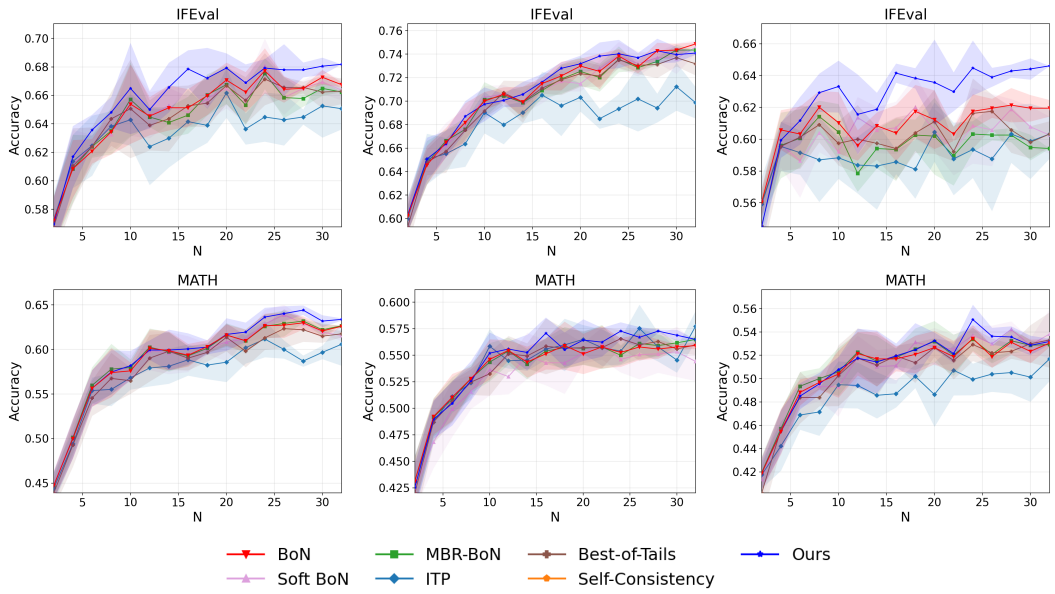


Figure 6: Best-of-N accuracy of LESS and the compared methods on `IFEval`, and `MATH` across different values of N. The first, second, and third columns use Skywork-Reward-V2-Qwen3-1.7B, Skywork-Reward-V2-Llama-3.2-3B, and Ai2 Llama-3.1-Tulu-3-8B-RM as the reward model, respectively.

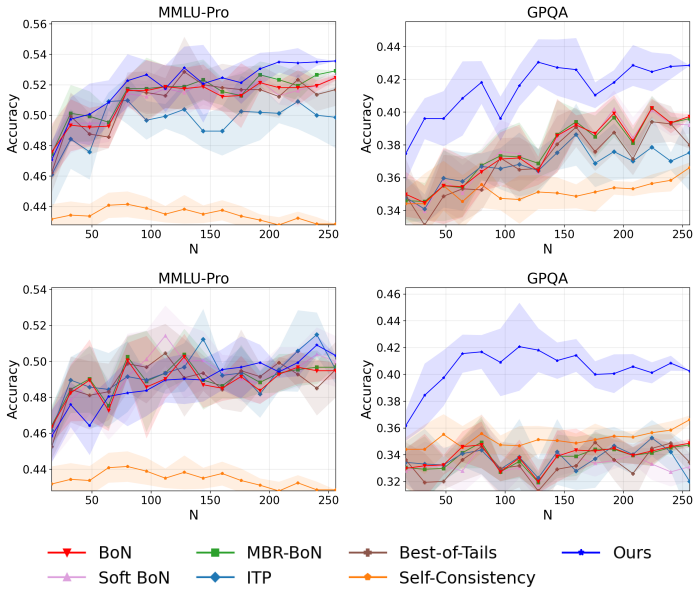


Figure 7: Best-of-N accuracy of LESS and the compared methods on `MMLU-Pro` and `GPQA` in the large-N regime. The first row uses Skywork-Reward-V2-Llama-3.2-3B as the reward model, while the second row uses Ai2 Llama-3.1-Tulu-3-8B-RM.

For each benchmark, `PPE` contains up to 512 prompt-level candidate sets. Each candidate set contains 32 independently sampled responses from a single generator. To avoid trivial selection instances, `PPE` filters out prompts for which all responses are correct or incorrect, and generally retains candidate sets with a correctness rate between 10% and 90%. Each response is assigned a binary correctness label using the original benchmark-specific verification procedure.

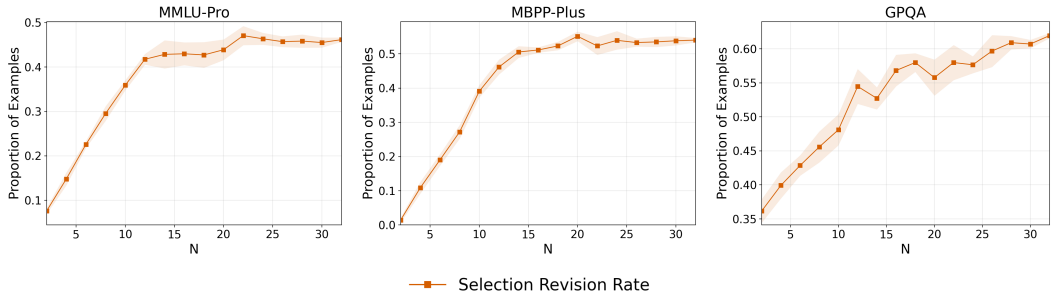


Figure 8: Selection revision rate of LESS relative to standard BoN on `MMLU-Pro`, `MBPP-Plus`, and `GPQA`, using Skywork-Reward-V2-Qwen3-1.7B as the reward model.

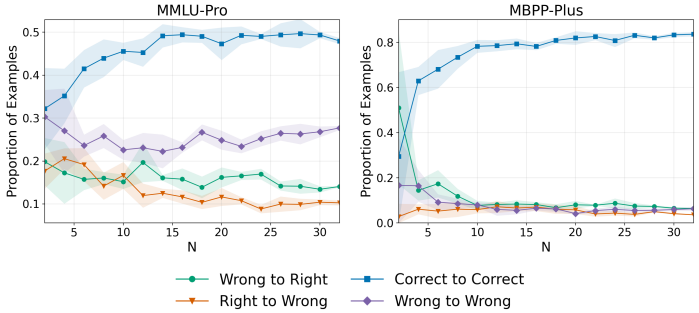


Figure 9: Correctness transition proportions from standard BoN to LESS on `MMLU-Pro` and `MBPP-Plus`, using Skywork-Reward-V2-Qwen3-1.7B as the reward model.

C Further Analysis

Architectural Choices. We examine whether the effectiveness of LESS depends on the specific architecture used for cross-candidate contextualization. We replace the Set Transformer encoder with a DeepSets encoder (Zaheer et al., 2017), while retaining the same input features and set-level training objective. The DeepSets variant incorporates information from other candidates through shared transformations and mean aggregation, whereas the default Set Transformer uses attention-based interactions. Thus, both variants retain cross-candidate contextualization. This comparison evaluates different interaction mechanisms, complementing the LESS-NS ablation that removes the set-wise encoder. As shown in Figure 10a, the two architectures achieve comparable performance, with the default configuration showing a modest advantage. These results suggest that LESS can benefit from different forms of cross-candidate contextualization without relying on a particular encoder architecture.

Objective Choices. We examine the effect of the training objective by replacing the set-level selection loss with a binary cross-entropy (BCE) loss, while retaining the same Set Transformer encoder and prediction head. In this variant, each candidate's logit is still computed from its contextualized representation and therefore depends on the other candidates. BCE applies a separate binary classification loss to each logit using the corresponding correctness label. It does not remove cross-candidate interactions. In contrast, our set-level objective jointly normalizes the candidate logits and maximizes the total probability assigned to candidates with the highest ground-truth reward, directly targeting response selection within the set. As shown in Figure 10b, the default set-level objective achieves slightly better performance than BCE. These results suggest that directly optimizing the probability of selecting an optimal candidate provides a modest benefit over supervising candidate correctness separately, even when both objectives use the same contextualized representations.

Ablation of Reward-Statistic Features. We further conduct a more fine-grained ablation of the reward-statistic features. Starting from the full reward features, we separately remove the gap from the highest-reward candidate, the relative reward rank, and the candidate-set size feature. The results

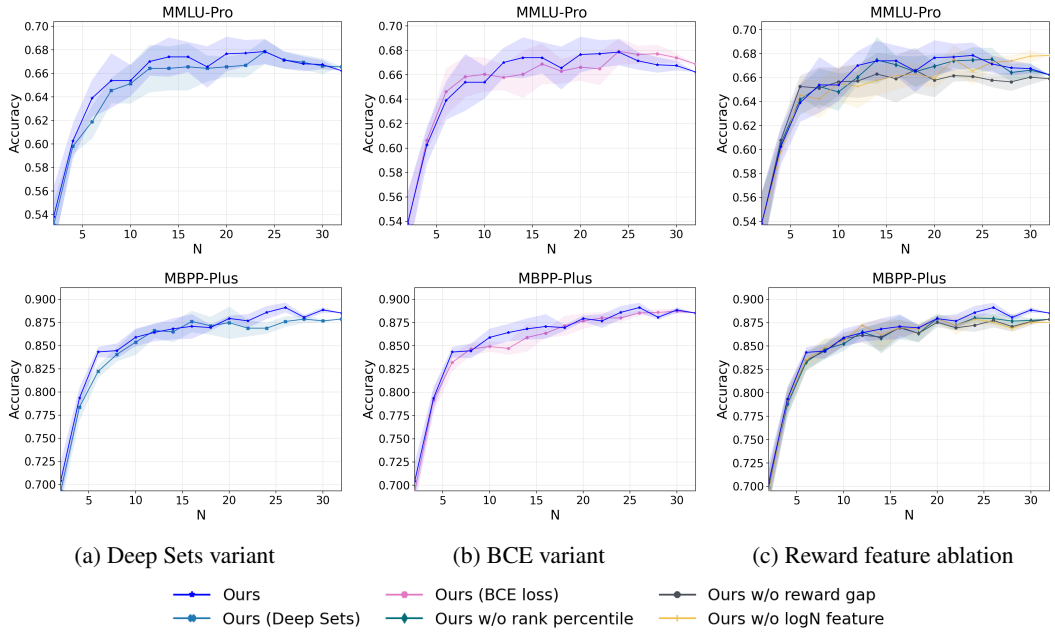


Figure 10: Best-of-N accuracy of LESS and its variants on `GPQA` across different values of N with Skywork-Reward-V2-Qwen3-1.7B used as the reward model.

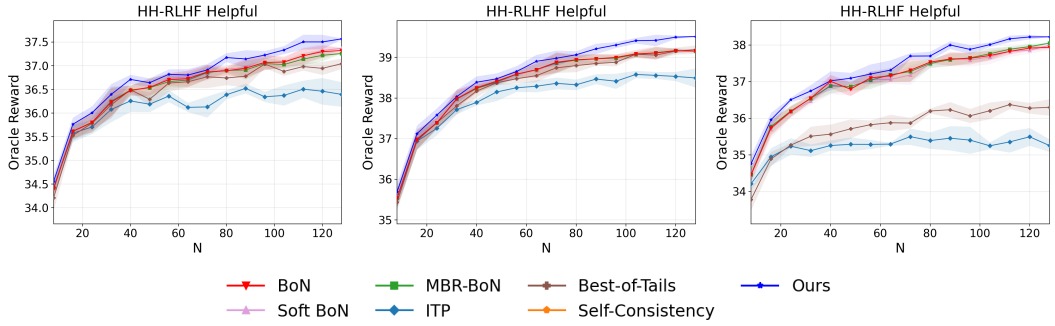


Figure 11: Oracle reward achieved by LESS and the compared methods on `Helpfulness` across different values of N. The left, middle, and right panels use Skywork-Reward-V2-Qwen3-1.7B, Skywork-Reward-V2-Llama-3.2-3B, and Ai2 Llama-3.1-Tulu-3-8B-RM as the proxy reward model, respectively. Skywork-Reward-V2-Llama-3.1-8B-40M is used as the oracle reward model.

in Figure 10c show that using the complete reward-statistic features generally achieves the best performance, while removing an individual feature may lead to a small degradation. These results indicate that the different reward statistics may provide complementary information for robust set-wise selection. Importantly, how these signals should be combined and weighted is learned directly by the selector from data, rather than specified through hand-crafted rules.

Evaluation on Open-Ended Benchmark. To evaluate whether LESS generalizes beyond tasks with verifiable correctness labels, we conduct an additional experiment on the `Helpfulness` subset of `HH-RLHF` (Bai et al., 2022), which contains open-ended dialogue prompts. We sample 500 prompts and use Llama-3-8B-Instruct to generate 128 candidate responses for each prompt. Following prior work (Huang et al., 2025; Jinnai et al., 2025), we employ Skywork-Reward-V2-Llama-3.1-8B-40M (Liu et al., 2025a), a strong and competitive reward model with leading performance on preference evaluation benchmarks, as the oracle reward model for assessing response helpfulness. As in our main experiments, we use 200 prompts and their associated candidate pools to train LESS, with the candidate receiving the highest oracle reward serving as the selection target, and reserve the

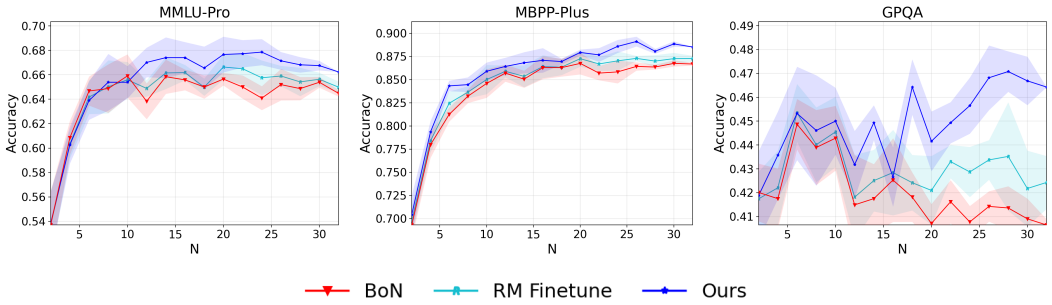


Figure 12: Best-of-N accuracy of LESS and the compared methods on `MMLU-Pro`, `MBPP-Plus`, and `GPQA` across different values of N with Skywork-Reward-V2-Qwen3-1.7B used as the reward model.

remaining 300 prompts for evaluation. As shown in Figure 11, LESS achieves consistently higher oracle reward for the selected responses across different candidate-set sizes. These results demonstrate that the effectiveness of LESS extends beyond benchmarks with binary correctness labels to open-ended preference-based selection.

Comparison with Reward-Model Fine-Tuning. One natural alternative to learning a separate set-wise selector is to directly adapt the reward model using the same setting-specific supervision. To investigate whether the gains of LESS can be simply attributed to access to additional labeled data, we fine-tune Skywork-Reward-V2-Qwen3-1.7B using the same 200 training prompts used for training LESS. Specifically, we apply LoRA adaptation (Hu et al., 2022) while jointly updating the reward head, and train the reward model using the same set-level objective and varying-size candidate-set subsampling protocol as LESS. The fine-tuned reward model is then directly used for standard Best-of-N selection, without introducing any additional calibration or selection module. As shown in Figure 12, fine-tuning the reward model improves its selection quality, but its performance still remains below that of LESS across different values of N. These results suggest that directly adapting the proxy reward model with limited setting-specific supervision is insufficient to fully address the reward errors exposed by Best-of-N selection. In contrast, LESS explicitly models the relative structure and interactions among candidates within each response set, allowing it to more effectively exploit the same amount of supervision for inference-time selection.

Cross-Setting Generalization and Adaptation. To examine whether a selector trained in one setting can generalize to other settings, we consider two transfer scenarios. First, we study cross-reward-model transfer, where the training and evaluation datasets remain the same but different reward models are used to provide the proxy rewards. Second, we study cross-dataset transfer, where the same reward model is used for training and evaluation but the datasets differ. As shown in Figures 13 and 14, under reward-model transfer, LESS retains relatively strong performance, although it performs worse than a selector trained directly with the corresponding evaluation-time reward model. In contrast, direct transfer across datasets results in a more noticeable performance degradation. This behavior is *consistent* with the design of LESS, which aims to learn a tailored selection strategy for an inference condition rather than a universally transferable selection rule. Different settings may exhibit different candidate characteristics and reward-error patterns, and fixed-rule approaches likewise typically require setting-specific hyperparameter tuning when the inference setting changes. We therefore further investigate whether LESS can efficiently adapt to a new dataset with limited target-setting supervision. Using only 50 samples from the target setting for fine-tuning, LESS recovers a large portion of its in-setting performance, as shown in Figure 15, suggesting that the learned selector can be efficiently adapted to new settings with modest supervision.

Computational Cost Analysis. Compared with standard Best-of-N, the additional computation introduced by LESS consists of two components: extracting prompt-response embeddings and running the learned set-wise selector. Embedding extraction constitutes the main computational overhead. For 100 prompts with 32 candidate responses per prompt (3,200 prompt-response pairs in total), computing the embeddings takes approximately 2 minutes with a peak GPU memory usage of about 9 GB. In contrast, the selector itself is lightweight. Training LESS on 200 prompts with

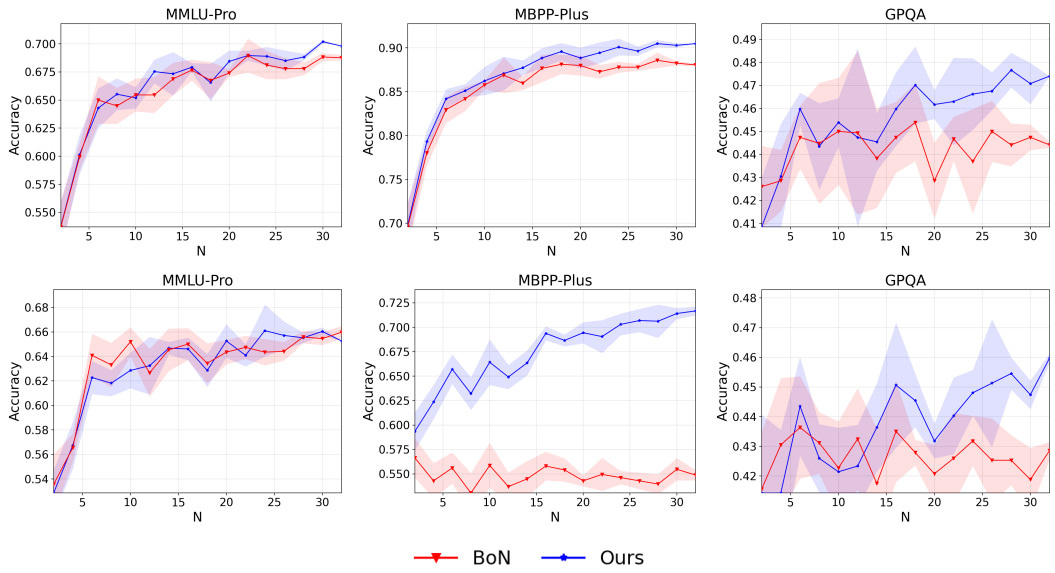


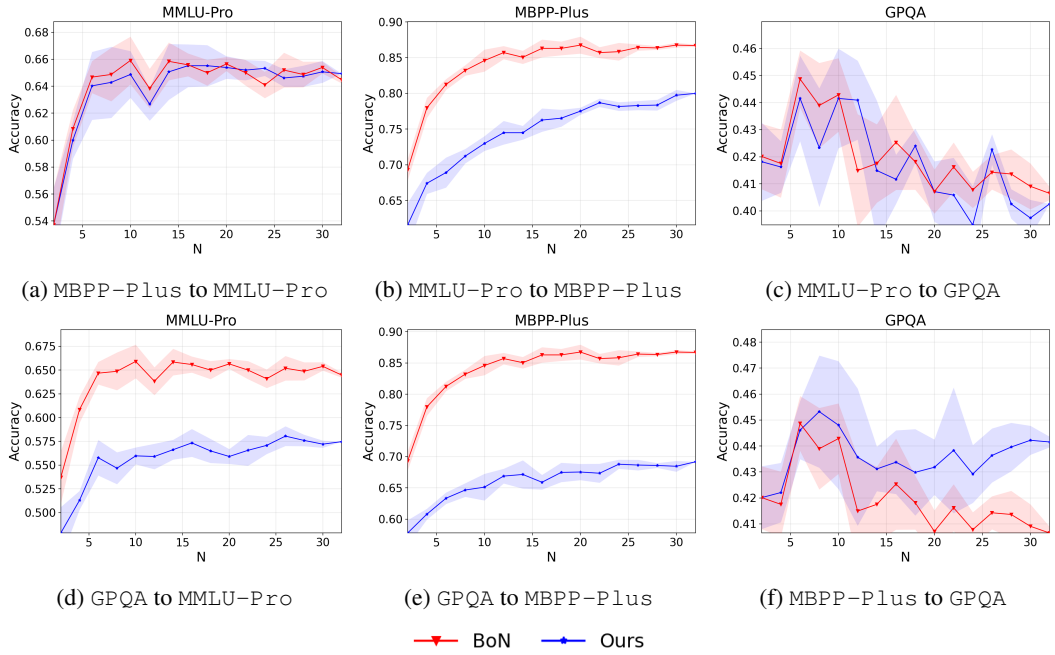
Figure 13: Best-of-N accuracy of LESS on `MMLU-Pro`, `MBPP-Plus`, and `GPQA` across different values of N under reward-model shift. LESS is trained with Skywork-Reward-V2-Qwen3-1.7B, and evaluated using rewards from Skywork-Reward-V2-Llama-3.2-3B (first row) and Ai2 Llama-3.1-Tulu-3-8B-RM (second row).

varying candidate-set sizes up to $N = 32$ takes approximately 8 seconds, while evaluation on 300 prompts across the considered candidate-set sizes takes approximately 6 seconds. In the large-N setting, with candidate-set sizes up to $N = 256$, training takes approximately 110 seconds and evaluation takes approximately 16 seconds. The GPU memory usage is similar between training and evaluation, requiring about 490 MB for $N \leq 32$ and about 810 MB for $N \leq 256$. Since the embedding model is frozen, these embeddings can also be precomputed and reused during selector training. Overall, the computational overhead of LESS is dominated by embedding extraction, whereas the additional cost of the set-wise selector is relatively small. All measurements are obtained on a single NVIDIA RTX 6000 Ada GPU.

Supervision Requirements. Existing methods also require setting-specific data, although the type and use of this data differ. Jinnai et al. (2025) use 1,000 development prompts in their main experiments to select a scalar regularization coefficient, while Yu et al. (2026) train an auxiliary predictor on up to 5,000 unlabeled prompt-response pairs. LESS is trained on 200 labeled prompt-level candidate pools. Because these methods use different forms of supervision, the raw example counts are not directly comparable. In our controlled evaluation, all methods are given access to the same 200-prompt development split for training or hyperparameter selection.

D Case Study

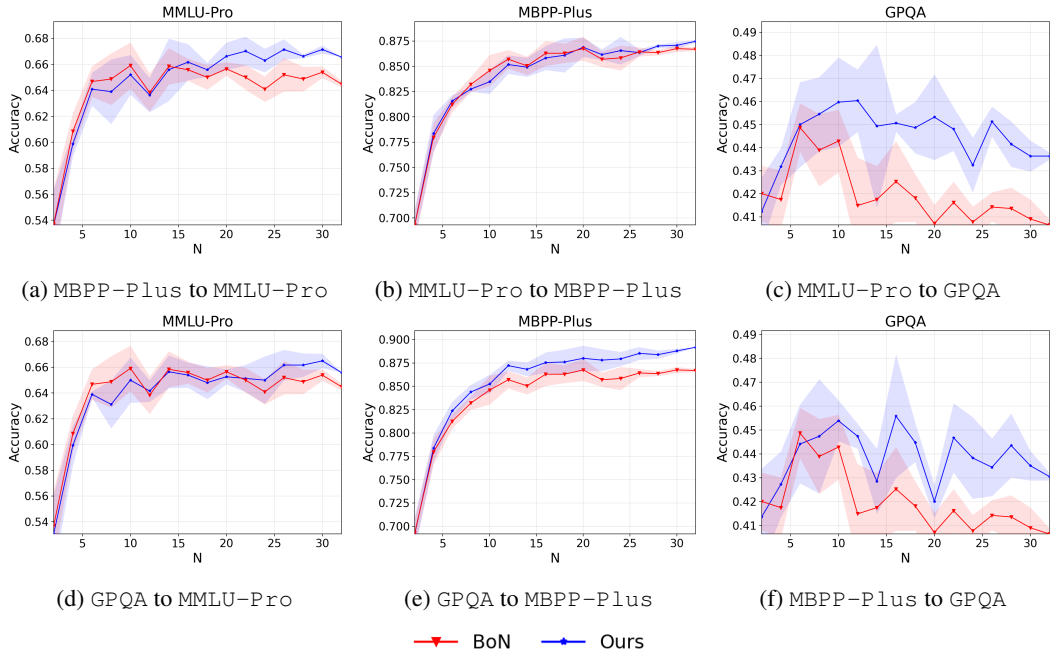
We present four case studies with $N = 8$ and $N = 16$ to examine how LESS changes response selection relative to standard Best-of-N. Figure 16, 17, 18, and 19 cover code generation, instruction following, and scientific question answering with two different reward models. Each figure compares the ordering induced by the original proxy rewards with that induced by the learned selection scores, and displays the response selected by LESS. Correctness is determined by the benchmark labels. In Figure 16, the two ground-truth-correct responses are ranked second and third according to the original reward scores. After applying LESS, they move to the first and second positions under the learned selection scores, allowing the correct response A2 to be successfully selected. In Figure 18, although the incorrect response A1 receives the highest original reward score, it is demoted to fourth under the learned selection scores. In contrast, the top three positions are all occupied by correct responses. These cases illustrate that LESS can effectively adjust the candidate



(a) MBPP-Plus to MMLU-Pro (b) MMLU-Pro to MBPP-Plus (c) MMLU-Pro to GPQA

(d) GPQA to MMLU-Pro (e) GPQA to MBPP-Plus (f) MBPP-Plus to GPQA

Figure 14: Best-of-N accuracy of LESS on MMLU-Pro, MBPP-Plus, and GPQA across different values of N under dataset shift, using Skywork-Reward-V2-Qwen3-1.7B as the reward model.

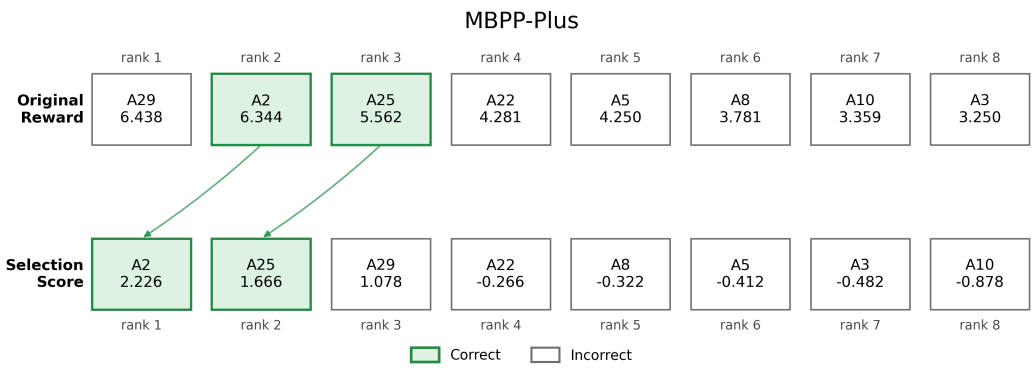


(a) MBPP-Plus to MMLU-Pro (b) MMLU-Pro to MBPP-Plus (c) MMLU-Pro to GPQA

(d) GPQA to MMLU-Pro (e) GPQA to MBPP-Plus (f) MBPP-Plus to GPQA

Figure 15: Best-of-N accuracy of LESS on MMLU-Pro, MBPP-Plus, and GPQA across different values of N under dataset shift, with LESS fine-tuned on 50 samples from the target dataset and Skywork-Reward-V2-Qwen3-1.7B used as the reward model.

ordering induced by an imperfect proxy reward model, promoting genuinely correct responses that may otherwise be overlooked and thereby improving the reliability of Best-of-N alignment.



Prompt:

```
Below will be an instruction to write a python function that
    accomplishes a task. You will also be given starter code with a
    function definition and any required imports. Think step-by-step
    , write comments clearly explaining each part of the code, and
    make sure your code solution is enclosed in markdown ticks (```
    [your code here] ```).

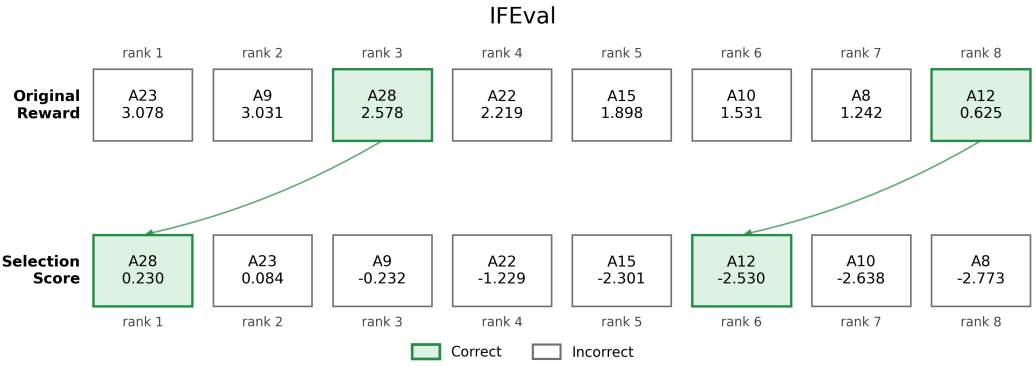
<instruction>
Write a function to find the closest smaller number than n.
</instruction>

<starter_code>
```
def closest_num(N):
    """
    Example Outputs:
    closest_num(11) == 10
    closest_num(7) == 6
    closest_num(12) == 11
    """
    pass
```
</starter_code>
```

Answer A2:

```
```python
def closest_num(N):
    """
    Example Outputs:
    closest_num(11) == 10
    closest_num(7) == 6
    closest_num(12) == 11
    """
    # Subtract 1 from N to explore the previous number
    closest = N - 1
    # Return the closest smaller number
    return closest
```
```

Figure 16: Case study of LESS on `MBPP-Plus` for $N = 8$, with Ai2 Llama-3.1-Tulu-3-8B-RM used as the reward model.



Prompt:

```
It is extemely important that you follow all instructions exactly:
Rewrite the following sentence into an email, and make sure it
    contains at least 10 placeholders represented by square brackets
    , such as [name]: The boots are warm but a little uncomfortable.
```

Answer A28:

```
Subject: Feedback on [Product Name] Boots

Dear [Recipient Name],

I'm writing to provide some feedback on the [Product Name] boots
    that I recently purchased on [Date of Purchase].

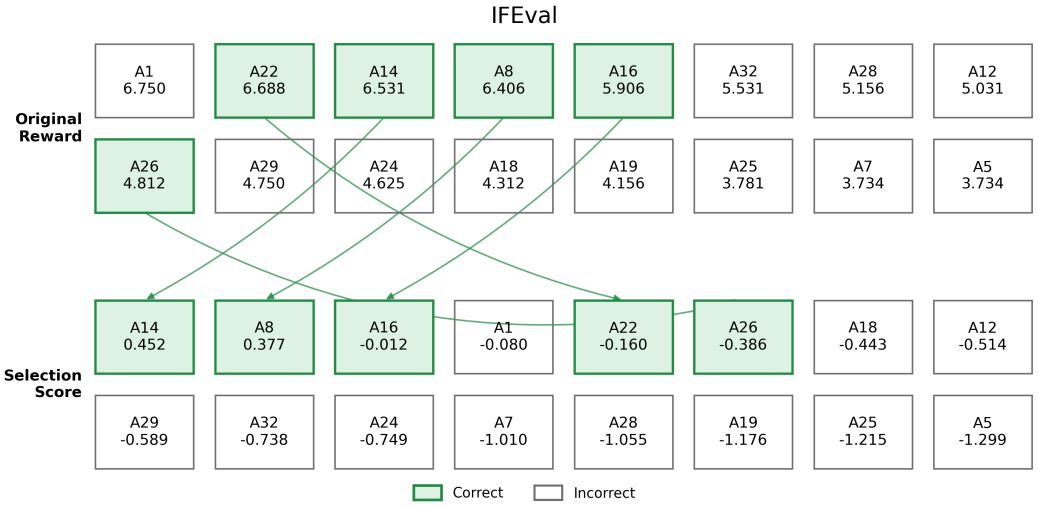
Overall, I'm satisfied with the boots. They are [warm/not warm] and
    appear to be well-made. However, I have found them to be [
    slightly uncomfortable/very uncomfortable]  [Explain where they
    are uncomfortable - e.g., around the toes, ankle, etc.].

I was hoping to [Get advice on solving the problem, e.g., exchange
    them for a different size, find a solution for the discomfort,
    etc.]. Could you please [Ask for advice or suggestion]?


Sincerely,

[Your Name]
```

Figure 17: Case study of LESS on `IfEval` for $N = 8$, with Skywork-Reward-V2-Qwen3-1.7B used as the reward model.



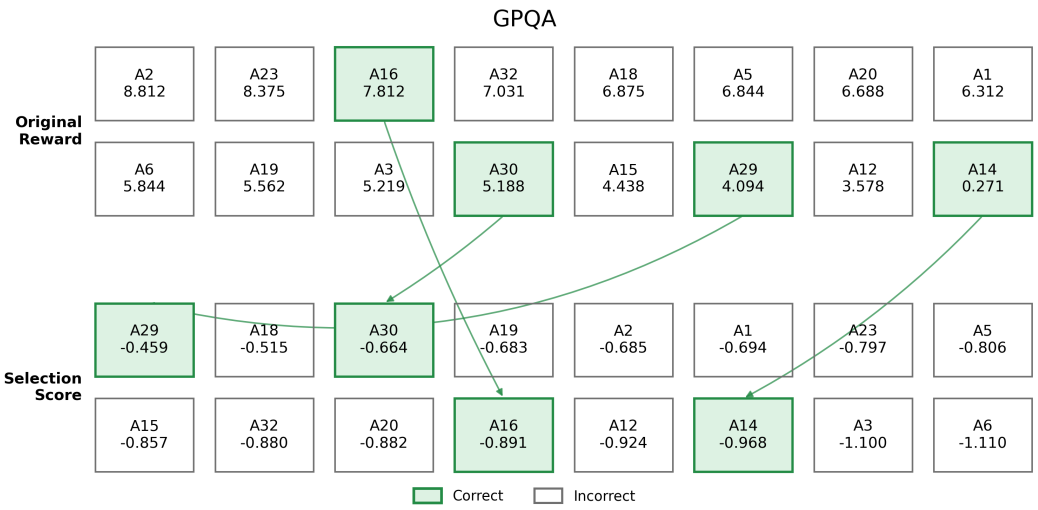
Prompt:

```
It is extemely important that you follow all instructions exactly:
Write a JSON schema for a beta cruiser that includes at least one
    placeholder represented by square brackets. The response must be
     in English and all lowercase letters.
```

Answer A14:

```
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "class": {
      "type": "string"
    },
    "manufacturer": {
      "type": "string"
    },
    ...
    "max_speed": {
      "type": "number"
    },
    "[placeholder]": {
      "type": "string"
    }
  },
  "required": [
    "name",
    "class",
    "manufacturer",
    ...
    "max_speed"
  ]
}
```

Figure 18: Case study of LESS on `IfEval` for $N = 16$, with Ai2 Llama-3.1-Tulu-3-8B-RM used as the reward model. [...] indicates omitted content for brevity.



Prompt:

```
The following is a chemistry multiple choice question. Think step by
     step and then finish your answer with "the answer is (X)" where
     X is the correct letter choice.

Question: Ozonolysis of compound A produces 3-methylcyclopentanone
    and acetone. Determine the product of the reaction of A with
    sulfuric acid under heat.
Choices:
(A) 1,2,4-trimethylcyclohex-1-ene
(B) 1-methyl-3-(propan-2-ylidene)cyclopentane
(C) 3,5,5-trimethylcyclohex-1-ene
(D) 1-isopropyl-3-methylcyclopentanol
```

Answer A29:

```
Here's the step-by-step solution:
1. **Understand Ozonolysis:** Ozonolysis cleaves carbon-carbon
    double bonds, typically forming carbonyl compounds (like ketones
     or aldehydes).
2. **Analyze the Products:**  The ozonolysis products are 3-
    methylcyclopentanone and acetone. This tells us that compound A
    must have contained a double bond that, when cleaved, formed
    these two ketones.
3. **Reconstruct Compound A:**  By working backward from the
    products, we can deduce the structure of compound A.  3-
    methylcyclopentanone suggests a cyclopentene ring with a methyl
    group at position 3. Acetone indicates a methyl group must have
    been present at one end of the double bond in A.
4. **Sulfuric Acid Reaction:** Sulfuric acid under heat is a strong
    dehydrating agent. It promotes the formation of alkenes via the
    elimination of water from alcohols.
5. **Predict the Product:** Since we know compound A contains a
    cyclopentene ring and a methyl group, reacting it with sulfuric
    acid will likely cause the elimination of water, leading to the
    formation of a more substituted alkene.
Considering these points, the most likely product is **(A) 1,2,4-
    trimethylcyclohex-1-ene**.
**The answer is (A)**.
```

Figure 19: Case study of LESS on GPQA for $N = 16$, with Skywork-Reward-V2-Qwen3-1.7B used as the reward model.