Check Your Work: Structured Checklist Feedback for Improving Large Language Models

Jonathan Cook
FLAIR, University of Oxford
jonathan.cook2@hertford.ox.ac.uk

Tim Rocktäschel
UCL AI Centre
t.rocktaschel@cs.ucl.ac.uk

Jakob Foerster
FLAIR, University of Oxford
jakob@robots.ox.ac.uk

Dennis Aumiller*
Cohere
dennisaumiller@cohere.com

Alex Wang*
Cohere
alexwang@cohere.com

Abstract

Much recent progress in Large Language Model (LLM) performance has been driven by verifiable feedback in deterministic domains like mathematics and code. However, scaling reinforcement learning (RL) and test-time compute in domains for which strict verification is infeasible remains a challenge. A common approach is to use an LLM-as-judge, which often relies on opaque, monolithic scores. In this work, we propose that AI feedback is most effective when decomposed into granular, prompt-specific checklists. To transform these checklists into a scalar reward, we introduce DIVA: DIscriminative VAriance weighting, a dynamic aggregation scheme that prioritises checklist items based on their ability to distinguish quality across a candidate pool. This ensures the reward signal focuses on the most salient criteria for a given prompt and response group, rather than being diluted by trivial or redundant constraints. Our approach yields an 11.8% win-rate improvement on AlpacaEval 2.0 using Qwen3-8B, outperforming holistic reward models and existing checklist baselines. Beyond training, we show that these checklists serve as a structured policy improvement operator at inference time — by using the model's own checklist evaluation as localised contextual feedback, the model can iteratively refine its output. This self-correction mechanism outperforms free-form sequential self-correction, offering a unified and interpretable framework for scaling both training-time and test-time performance in domains lacking strict verifiers.

1 Introduction

Instruction-tuned Large Language Models (LLMs) are widely used as conversational assistants (Wei et al., 2022; Mishra et al., 2022; Bai et al., 2022a; Ouyang et al., 2022). In domains with deterministic verifiers, such as mathematics and code, the ability to provide ground-truth feedback has enabled significant performance leaps. In these verifiable regimes, techniques such as rejection sampling (Snell et al., 2024), Reinforcement Learning (RL) (Lambert et al., 2024a), and in-context refinement (Madaan et al., 2023; Shinn et al., 2023) allow models to iteratively bridge the gap between initial attempts and correct solutions.

However, many real-world tasks, such as writing and open-domain question answering, lack deterministic verifiers. In these cases, AI feedback has emerged as a practical alternative (Zheng et al., 2023). For training purposes, this feedback often relies on holistic preference scores that are opaque and difficult to interpret (Qin et al., 2024). For test-time refinement, existing strategies typically rely on "free-form" self-correction, where a model generates a qualitative critique of its own output (Madaan et al., 2023; Shinn et al., 2023). While flexible, free-form feedback often lacks the precision required for consistent improvement; models frequently fail to identify specific errors or suffer from "sycophancy," where they overlook flaws they themselves generated (Huang et al., 2024; Kamoi et al., 2024). These limitations hinder the reliability of AI feedback across both training-time RL and inference-time refinement.

To address these challenges, we propose that AI feedback is most effective when it is decomposed into binary, verification-like steps. We decompose evaluation tasks into granular, prompt-specific YES/NO checklists, transforming a subjective judgement into a set of discrete constraints. To effectively aggregate these binary signals for training, we introduce DIVA: DIscriminative VAriance weighting, a dynamic scheme that prioritises checklist items based on their variance across a candidate pool. Unlike prior work that relies on static importance weights — which are themselves subject to model bias — DIVA uses the data-driven distribution of responses to identify the most salient

*Equal advising.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16649–16688

criteria. Furthermore, for test-time improvement, these checklists provide a structured feedback loop that is significantly more actionable than free-form prose. By grounding the refinement process in explicit binary constraints, we enable the model to systematically address its own failures.

We explore the utility of checklists across two primary axes of LLM optimisation:

1. As a reward function for RL finetuning: DIVA provides an effective reward signal for RL on general instruction sets without requiring ground-truth targets. Using `Llama-3.3-70B-Instruct` (Dubey et al., 2024) as a judge, RL finetuning of `Qwen3-8B` (Yang et al., 2025) on OpenAssistant (Köpf et al., 2023) instructions achieves an 11.8% win-rate improvement on AlpacaEval 2.0 (Li et al., 2023a). This significantly outperforms holistic AI feedback (RLAIF) (Lee et al., 2024), standard reward models (RLHF) (Ouyang et al., 2022), and checklist methods using static importance weights (RLCF) (Viswanathan et al., 2025).

2. As an inference-time policy improvement operator: Beyond training, checklists serve as interpretable feedback for response refinement. By using a model's own checklist evaluations as localised contextual feedback, the model can identify and rectify specific failures. This mechanism yields gains of up to 7.5% and 6.4% on InFoBench (Qin et al., 2024) and WildBench V2 (Lin et al., 2024) respectively, outperforming free-form self-refinement.

Finally, we validate the reliability of checklist feedback through extensive meta-evaluations, showing higher alignment with human judgement than holistic scoring and demonstrating that LLM-generated checklists can even improve inter-annotator agreement in human evaluation.

Our contributions are as follows:

1. We introduce a novel variance-based weighting scheme that dynamically identifies discriminative checklist items for more precise AI feedback. This yields improvements over key baselines in RL finetuning.

2. We formalise the use of checklists as a policy improvement operator, enabling more effective test-time refinement than free-form feedback.

3. We provide empirical evidence that checklist-based evaluation is more reliable and consistent for both AI and human judges.

2 Related Work

AI feedback (i.e., feedback extracted from a generative LLM) is increasingly used for evaluating LLM outputs (Li et al., 2023b; Lin et al., 2024; Zheng et al., 2023), due to its flexibility and low cost. This has also enabled AI feedback to be used for deriving rewards for RL (Bai et al., 2022b; Kwon et al., 2023; Roit et al., 2023; Yang et al., 2023; Lee et al., 2024; Yu et al., 2024). Lee et al. (2024) find that RL from AI Feedback (RLAIF) can yield superior alignment results to RLHF when fine-tuned models are evaluated by humans. They also show that direct-RLAIF (i.e., having an LLM directly generate a score) is more performant than preference-based RLAIF (i.e., having an LLM generate preference labels for training an RM). Together, these results indicate that the staleness of an RM that has been trained on a fixed preference dataset can limit its effectiveness when used downstream. Our RL fine-tuning results support this finding and show that checklists further enhance AI feedback.

Pan et al. (2024) conduct a study with human participants to explore design choices for AI feedback systems. Participants expressed a desire for *structured and case-specific evaluations* and highlighted the need for *distinct evaluation criteria for various tasks*. Structure and task-specificity are the core aspects of checklist feedback, both of which contribute to its effectiveness as in-context feedback, rewards, or in assisting human evaluators.

Several studies show that natural language critiques can be used to improve the quality of evaluations performed by a judge LLM or RM (Ankner et al., 2024; Bai et al., 2022b; Wang et al., 2023; Ye et al., 2024; Sun et al., 2024), inform human evaluations (Saunders et al., 2022; McAleese et al., 2024), or provide feedback that can be used to refine a response in-context (Scheurer et al., 2023; Tian et al., 2024; Madaan et al., 2023; Yuan et al., 2024). Meanwhile, a number of papers provide evidence that naively prompting LLMs to self-correct or find reasoning errors can lead to performance degradation (Huang et al., 2024; Tyen et al., 2024; Kamoi et al., 2024). Other work finetunes models specifically for self-correction, such as via RL (Kumar et al., 2024; Gehring et al., 2024). In comparison, we consider how structured feedback can overcome the pitfalls of prompted self-correction.

A number of fixed benchmarks for evaluating constrained instruction-following in LLMs decompose instructions into checklists (Zhou et al., 2023;

Jiang et al., 2024; Qin et al., 2024; Wen et al., 2024). These benchmarks pair their instruction data with curated checklists that involve manual effort to produce. Approaches to evaluation used in these works are therefore hard to make use of outside of the benchmarks themselves. We instead use an LLM to generate checklists, allowing them to be used on-the-fly to provide rewards or in-context feedback.

Recent work on Rubrics as Rewards (RaR) (Gunjal et al., 2025) uses instance-specific rubrics as reward functions for reinforcement learning, primarily in domain-specific expert reasoning tasks. While RaR demonstrates that rubric-based rewards can be effective, it relies on static or implicitly aggregated rubric criteria. In contrast, our results show that how checklist items are aggregated — specifically, dynamically prioritising discriminative criteria — is critical for performance, even when using the same underlying feedback structure. Closely related is Reinforcement Learning from Checklist Feedback (RLCF) (Viswanathan et al., 2025), which shows that checklist-based supervision can outperform learned reward models and holistic AI judges. Our work builds on this insight but differs in two key ways: we identify criterion aggregation as a central design problem, and we show that static weighting can underperform relative to variance-based prioritisation. Additionally, unlike RLCF, we demonstrate that checklists can act as a policy improvement operator at inference time, enabling structured self-refinement without further training. Together, these works highlight the promise of structured feedback, while our results suggest that the effectiveness of checklists depends on how their components are weighted and acted upon.

3 Methods

The core of our approach is the decomposition of an arbitrary instruction-following evaluation task into a structured, granular format. We define a checklist $\mathcal{C}_x = \{c_1, \ldots, c_M\}$ for a given prompt x, where each c_j is a binary question (e.g., "Does the response use a professional tone?"). This formulation closely mirrors item response theory (IRT) (Embretson and Reise, 2013) from psychometrics, which models how a latent ability interacts with individual test items to produce observable responses. We exploit this connection to design more informative reward signals and refinement operators for LLMs.

In our setting, the latent variable corresponds to a model's instruction-following competence for a given prompt, while checklist items act as evaluation "items" whose pass/fail outcomes reveal information about that competence. Crucially, IRT suggests that not all items are equally informative: items that fail to distinguish between high- and low-quality responses provide little learning signal. Our method operationalises this principle directly.

3.1 Checklists as Rewards (RL Training)

In classical Item Response Theory (IRT), each item is associated with a discrimination parameter that reflects how strongly performance on that item correlates with a latent ability. Highly discriminative items are those whose outcomes vary systematically across individuals and therefore provide the most information about ability, while items that everyone passes or fails contribute little signal.

Motivated by this insight, we treat the discriminative power of a checklist item as prompt- and policy-dependent, and estimate it directly from the empirical distribution of model outputs. We call this aggregation strategy **DIVA**: **DI**scriminative **VA**riance weighting.

For a prompt x, we generate a group of N candidate responses $\mathbf{y} = \{y_1, \ldots, y_N\}$. An LLM-as-judge evaluates each response y_i against the checklist $\mathbf{c}_x$. Rather than sampling a discrete `YES/NO` judgment, we use the judge's token-level logits to compute the probability that each checklist item is satisfied. This yields a soft score matrix $S \in [0,1]^{N \times M}$, where $s_{i,j}$ denotes the predicted probability that response y_i satisfies checklist item c_j.

For each checklist item c_j, we compute its mean success probability across the group and its corresponding empirical variance:

$$p_j = \frac{1}{N}\sum_{i=1}^{N} s_{i,j}, \quad v_j = \frac{1}{N}\sum_{i=1}^{N}(s_{i,j} - p_j)^2. \quad (1)$$

This variance is maximised when responses are evenly split between high and low predicted success and vanishes when the judge assigns consistently high or low probabilities across the group. As in IRT, this captures the extent to which an item differentiates responses at the current policy. Importantly, using soft probabilities reduces quantisation noise and allows variance arising from confident discrimination to be distinguished from variance arising from judge uncertainty.

Rather than learning static item parameters, we estimate discrimination online, conditioned on the current policy and prompt. We compute the weight w_j via normalisation across items with a smoothing parameter ϵ used for numerical stability (set to 1e-8 in practice):

$$w_j = \frac{v_j + \epsilon}{\sum_{k=1}^{M}(v_k + \epsilon)}. \tag{2}$$

We integrate DIVA into Group Relative Policy Optimisation (GRPO) (Shao et al., 2024), where the advantage of a response is computed relative to the mean reward of its group. The reward for a response y_i is defined as the weighted sum of its checklist scores:

$$R(y_i, x) = \sum_{j=1}^{M} w_j \cdot s_{i,j}. \tag{3}$$

DIVA prevents the reward from being dominated by criteria that the model already satisfies across most candidates, instead dynamically emphasising checklist items that lie at the current frontier of the policy's capabilities. Using probability-based checklist scores further improves robustness by attenuating the influence of items that are evaluated with high uncertainty — checklist items for which the judge assigns similar, low-confidence probabilities across responses induce low effective variance and therefore receive less weight. While variance-based weighting cannot recover signal that is entirely destroyed by judge noise, we find empirically that probability-based checklist feedback exhibits higher agreement with human judgments than direct scores and preferences (Section 4.3).

3.2 Checklists for Policy Improvement (Inference)

Beyond training-time rewards, the IRT framing also motivates a structured approach to inference-time refinement. In educational settings, failed items are those whose difficulty exceeds the learner's current ability (Embretson and Reise, 2013); targeted practice on these items is known to be more effective than unguided revision (Rahimi, 2021). We apply the same principle to LLM self-improvement.

For a given prompt x and initial response $y^{(0)}$, the model generates its own checklist $\mathbf{c}_x$ and performs a self-evaluation $\mathbf{s}^{(0)} = \{s_1^{(0)}, \ldots, s_M^{(0)}\}$. Rather than producing a free-form critique, we construct a refinement prompt conditioned only on the checklist residuals (i.e., failed checklist items). The refined response $y^{(1)}$ is generated as:

$$y^{(1)} \sim \pi_\theta(\cdot|x, y^{(0)}, \{c_j \in \mathbf{c}_x : s_j^{(0)} = 0\}) \tag{4}$$

This turns checklist residuals into a localised policy improvement operator, explicitly targeting the criteria that the model has not yet deemed to be satisfied. Unlike free-form self-correction, this approach grounds sequential refinement in discrete, verification-like signals.

3.3 Using Generated Checklists in Human Evaluation

In spite of recent progress in automatic evaluation (Ankner et al., 2024; Chiang and Lee, 2023; Verga et al., 2024; Vu et al., 2024; Wang et al., 2024a,b; Ye et al., 2024), human evaluation remains a critical component of LLM quality assessment. However, human evaluation itself can yield unreliable results (Hosking et al., 2024). We thus investigate whether checklists can help with arriving at consistent scores and preference judgements. Using checklists partially decomposes the annotation task to be more cognitively feasible, and helps to ensure that specific considerations relevant to each instruction are not missed. Annotators are thus asked to first answer checklist questions generated by an LLM before providing a quality rating for an individual response, or a preference between two responses. They are instructed to use the checklists to inform their judgement as they deem appropriate. This allows us to assess how using checklists impacts the ratings and inter-annotator agreement obtained from human evaluations.

4 Experiments

4.1 Checklists as Rewards for RL

Training Details. For training, we use queries from the OpenAssistant dataset (Köpf et al., 2023), which comprises high quality human conversational data designed to resemble interactions between humans and conversational assistants. We use `Qwen3-8B` as the base model. Finetuning is performed via GRPO for a single epoch with a batch size of 128 and a learning rate of 1e-5. The group size is set to 16, the sampling temperature is set to 0.7, and we use a KL β of 0.05.

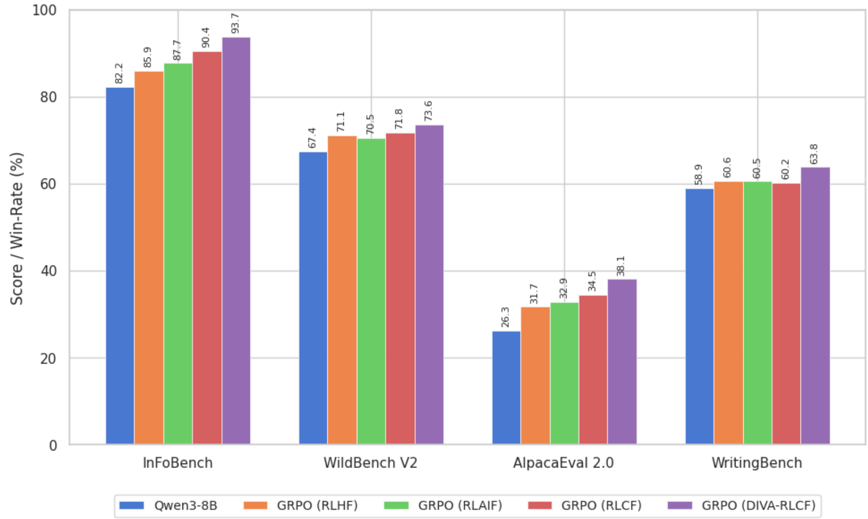

Figure 1: RL with DIVA-weighted checklist feedback leads to the greatest improvement to `Qwen3-8B` across all benchmarks.

Benchmarks. We evaluate on InFoBench (Qin et al., 2024), which measures instruction following under granular constraints; WildBench V2 (Lin et al., 2024) and AlpacaEval 2.0 (Li et al., 2023a), which comprise a broad range of instruction-following tasks designed to resemble real-world user queries; and WritingBench (Wu et al., 2025), which evaluates LLMs across 6 core writing domains and 100 subdomains.

Baselines. We compare our approach (DIVA-RLCF), using `Llama-3.3-70B-Instruct` as a judge, to the following baselines:

- **RLHF:** We take the reward model `Skywork-Reward-Gemma-2-27B` (Liu et al., 2024), which ranks competitively on RewardBench (Lambert et al., 2024b).
- **RLAIF:** We use the same judge model to give direct scores in the form of a 1-10 integer rating to each response. This follows the best practices for AI feedback established in Lee et al. (2024).
- **RLCF:** Following Viswanathan et al. (2025), we have the same judge model perform checklist evaluation and generate its own importance weights for each checklist item on a 1-100 scale.

Results Figure 1 shows that, across all benchmarks, RL fine-tuning with DIVA-weighted checklist rewards (DIVA-RLCF) consistently outperforms RLHF, RLAIF, and static-weight checklist baselines. On AlpacaEval 2.0, DIVA yields an 11.8% absolute win-rate improvement over the base `Qwen3-8B` model, substantially exceeding gains obtained with holistic AI feedback (RLAIF) and a strong learned reward model (RLHF). This confirms that decomposing feedback into granular constraints, and dynamically prioritising discriminative ones, produces a more effective training signal than monolithic scalar judgements.

Improvements are also pronounced on InFoBench, which explicitly evaluates fine-grained constraint satisfaction. Here, DIVA achieves the largest relative gains over all baselines, while static-weight checklist rewards underperform, indicating that naïvely weighting checklist items can dilute learning by over-emphasising constraints that are either trivial or universally satisfied. This gap high-

Method	Variance weighting	Soft judge scores	Win rate (%)
DRFR + sampling	✗	✗	32.9
DRFR + logits	✗	✓	33.6
MeanRec + sampling	✗	✗	33.3
MeanRec + logits	✗	✓	35.1
DIVA + sampling	✓	✗	35.4
DIVA + logits	✓	✓	**38.1**

Table 1: Ablation of checklist aggregation and judge modelling choices evaluated on AlpacaEval 2.0.

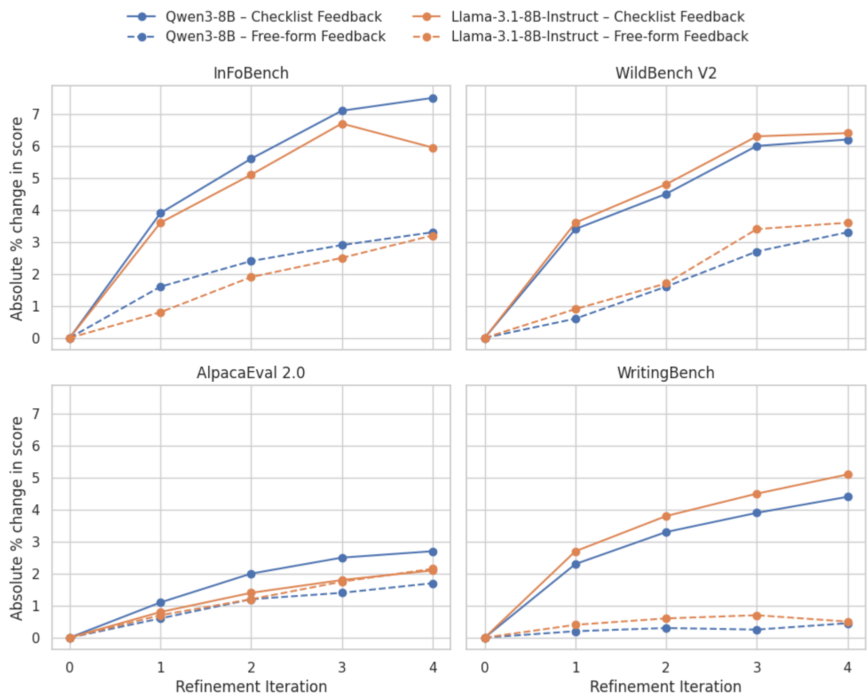

Figure 2: Iterated refinement of responses leads to significantly greater improvements on `Qwen3-8B` and `Llama-3.1-8B-Instruct` when self-generated feedback is in the form of checklist residuals rather than being free-form.

lights the importance of variance-based weighting for isolating learning-relevant criteria.

On WildBench V2 and WritingBench, DIVA consistently improves performance, demonstrating that the benefits of discriminative checklist rewards extend beyond synthetic constraint-following tasks to noisy, real-world user instructions and open-ended writing domains. In contrast, RLAIF shows smaller gains, suggesting that holistic scores provide a weaker and less targeted optimisation signal under GRPO.

Table 1 shows the impact of core design decisions for checklist evaluation and aggregation on final AlpacaEval 2.0 performance. We compare uniform (DRFR) (Qin et al., 2024) and mean-reciprocal weighting, which assigns higher weight to checklist items with low average success probability, versus variance-weighted (DIVA) aggregation and binary versus probability-based checklist scores. Both variance weighting and soft judge scores contribute to performance, with their combination yielding the strongest gains. Overall, these results show that DIVA enables checklist feedback to function as a robust and scalable reward function, outperforming both learned reward models and direct AI feedback.

4.2 Checklists as Policy Improvement Operators for Refinement

Setup We use the same benchmarks as in Section 4.1. The specific prompts for refining responses based on self-identified checklist failures and via free-form feedback can be found in Appendix G.

Results We evaluate checklist-based refinement against free-form self-refinement across the same benchmarks. Using checklist residuals as localised feedback consistently improves response quality relative to both the initial generation and unconstrained self-correction. On InFoBench, checklist-based refinement yields up to 7.5% absolute improvement, reflecting its ability to directly target unmet constraints identified during self-evaluation. On WildBench V2, we see up 6.4% absolute improvement, despite the benchmark's high variance and prevalence of ambiguous or underspecified instructions. This suggests that explicitly enumerating failures provides a stronger corrective signal than asking the model to "improve" its response in free-form prose. Qualitative inspection shows that free-form critiques often restate surface-level qualities or rationalise existing responses, whereas checklist residuals force concrete edits aligned with explicit criteria (see Appendix F.1).

Importantly, checklist-based refinement does not require any additional supervision or external judge at inference time — the model generates, evaluates, and acts upon its own checklist. Taken together, these results demonstrate that checklists are not only effective as training-time rewards but also serve as a reliable policy improvement operator at inference time, outperforming unstructured self-correction.

4.3 Meta-Evaluation of Checklist Feedback

To further investigate the effectiveness of using LLM-generated checklist feedback for LLM-generated responses, we compare generated checklist feedback to evaluations gathered from *trained*[1] *human annotators*. These annotations comprise pairwise preference evaluations and human-written checklist feedback. With these, we compare LLM-generated checklist feedback to equivalent, checklist-based human evaluations, as well as comparing to the more commonly used human preference evaluations.

Dataset & Annotations. We gather annotations for a set of 500 WildBench V2 instructions. We first collect preferences by providing annotators with a pair of responses from different models and asking them to indicate their preference on an integer sliding scale from 1, meaning *"Response A is much better than Response B"*, to 5, meaning the reciprocal strong preference (see Appendix H for further details). Each response pair is triply annotated and we compute the average preference score $\bar{p}$ across these three annotations. We then bin each average preference into a win ($1 \leq \bar{p} < 2.5$), tie ($2.5 \leq \bar{p} \leq 3.5$), or loss ($3.5 < \bar{p} \leq 5$). We also collect checklist-based annotations, where the annotators are given precise requirements for writing and answering checklists. To produce a preference judgement between two responses using generated checklist feedback, each response is scored via DIVA-weighted checklist rewards and the response with the higher reward is considered to be preferred.

We follow Qin et al. (2024) and use the Pairwise Label Distance (PLD) to measure agreement between checklist feedback and human preference labels. The Weighted Pairwise Label Distance (WPLD) is defined in Equation 5, where N is the number of instructions. The WPLD thus ranges from $0 \rightarrow 2$, with a lower value indicating stronger agreement.

[1]Details of the annotator training are in Appendix H.

Method	Pairwise Agreement w/ Humans			
	PLD-0	PLD-1	PLD-2	WPLD
Pref	0.309	0.497	0.194	0.885
Score	0.445	0.503	0.052	0.607
C-Score	0.498	0.465	0.037	0.539
C-DRFR	0.526	0.435	0.039	0.513
C-DIVA	0.562	0.407	0.031	**0.469**

Table 2: Agreement between different LLM-generated evaluations and pairwise preferences from trained human annotators on WildBench V2. GPT-4o is used as the judge LLM.

$$\text{WPLD} = \sum_{i=0}^{2} \frac{i}{N} \sum_{j=0}^{N} \mathbb{I}[\text{PLD}_j = i] \qquad (5)$$

Models & Baselines. We compare making preference judgements via DIVA-weighted checklists (C-DIVA) to uniformly aggregating checklist answers (C-DRFR) (Qin et al., 2024), directly prompting an LLM to express a preference, and direct scores (1-10 quality rating). We also consider a hybrid of checklist and direct feedback, C-Score, where generated checklists are included in the judge prompt, but judges are not required to explicitly answer each checklist question, instead providing a quality rating. Response pairs are formed out of generations from Command-R+ (Cohere, 2024), GPT-4o (OpenAI, 2024) and Claude-3-Sonnet (Anthropic, 2023). We use GPT-4o as the judge LLM.

Results. In Table 2, we see that checklist feedback aggregated via DIVA agrees most strongly with human preferences, achieving the lowest overall WPLD. Checklists are also the only evaluation strategy to achieve a PLD of 0 more often than not. Uniformly aggregated checklist feedback (C-DRFR) and C-Score agree more strongly with humans than vanilla feedback, which confirms the general utility of checklists in evaluation. However, the fact that C-Score falls short compared to aggregated checklist feedback suggests that the added structure from explicitly answering and combining checklist responses is valuable. Despite the fact that LLM-generated preference judgements are very common (Lee et al., 2024) prompting a judge LLM to directly produce a preference results in the lowest agreement with humans. Overall, these results show that *evaluations elicited from an LLM benefit from a structured and dynamically weighted* *scoring protocol*, even when that protocol is task-agnostic and generally applicable, as in the case of checklist feedback.

4.4 Assisting Human Evaluation

Setup. We conduct two rounds of human evaluation on the same set of Command-R+ and Claude-3-Sonnet responses to WildBench V2 instructions. In both rounds, annotators provide either an integer score from 1-5 for each Command-R+ response or a pairwise preference between the responses from each model (further details in Appendix H). However, in one round, they are first asked to answer checklist questions generated by GPT-4o before providing each score or preference (C-Score and C-Pref). They are instructed to use the checklists to inform their judgement as they see fit. Each evaluation is triply annotated and we compute Inter-Annotator Agreement (IAA) using Krippendorff's alpha (Krippendorff, 1980), because of its flexibility with handling ordinal evaluations for direct scoring and the random assignment of annotators to responses.

Results. Figure 3 shows the IAA among triply annotated labels for each protocol. These indicate that annotating a generated checklist before scoring yields stronger agreement among evaluators. We also find that the average direct score for Command-R+ across both settings stays remarkably consistent, with a difference of 0.004, implying that the increase in agreement corresponds to variance reduction without having a biasing effect on the evaluations. Finally, we see that annotators qualitatively find evaluating responses easier when using checklists (see Appendix H.4). Overall, these results highlight the potential for AI-assisted AI evaluation under a sufficiently structured judgement protocol.

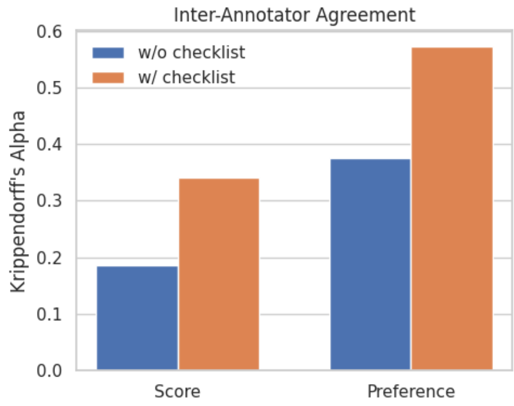

Figure 3: Inter-Annotator Agreement (Krippendorff's alpha) among triply annotated labels when providing 1-5 scores for Command-R+ responses or pairwise preferences between Command-R+ and Claude-3-Sonnet responses on WildBench V2.

5 Conclusion

In this work, we demonstrate that decomposing AI feedback into granular, prompt-specific checklists provides a more effective and interpretable framework for optimising LLMs. By moving away from opaque, monolithic scoring and towards structured binary evaluation, we address critical limitations in current AI feedback systems, such as the lack of precision in free-form self-correction and the biases inherent in static reward weighting.

We introduce DIVA, a data-driven approach to RL training that prioritises checklist items based on their ability to distinguish quality among candidate responses. This ensures that the training signal remains focused on the most salient criteria for improvement, yielding an 11.8% win-rate improvement on AlpacaEval 2.0 and outperforming existing holistic and static-weight reward models. Furthermore, we established that checklists can serve as a powerful policy improvement operator at inference time. By using a model's own checklist evaluations as localised contextual feedback, we enable a structured self-correction strategy that consistently outperforms free-form refinement. Finally, our meta-evaluations confirm that checklist-based feedback is more reliable and aligns more closely with human judgement than traditional scoring methods. By providing a unified framework for both training-time RL and test-time refinement, checklist feedback offers a path towards more generally capable AI assistants.

Limitations

While checklist feedback demonstrates clear benefits, it is not without limitations. Generating accurate and task-relevant checklists depends on the quality of the underlying LLM, which may introduce errors or overlook some evaluation criteria. Additionally, the reliance on binary YES/NO questions, while simplifying evaluation, may oversimplify very open-ended tasks requiring more nuanced judgements. Furthermore, while we account for model uncertainty by using logit-based success probabilities, checklist evaluations could still be subject to systematic errors on the part of the judge model. Using a jury of multiple LLMs could be a potential solution to this, but would significantly increase the cost. Future work should investigate ways to reduce the computational overhead of generating and evaluating checklist items to form a reward or refinement feedback.

References

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. 2024. Critique-out-Loud Reward Models. *arXiv preprint arXiv:2408.11791*.

Anthropic. 2023. The Claude 3 Model Family: Opus, Sonnet, Haiku.

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, and 12 others. 2022a. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022b. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*.

Cheng-Han Chiang and Hung-yi Lee. 2023. A closer look into using large language models for automatic evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8928–8942, Singapore. Association for Computational Linguistics.

Cohere. 2024. Command R+ Model.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Susan E Embretson and Steven P Reise. 2013. *Item response theory for psychologists*. Psychology Press.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. 2024. RLEF: grounding code llms in execution feedback with reinforcement learning. *CoRR*, abs/2410.02089.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *CoRR*, abs/2507.17746.

Tom Hosking, Phil Blunsom, and Max Bartolo. 2024. Human Feedback is not Gold Standard. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs. *arXiv preprint arXiv:2406.01297*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology*. SAGE Publications.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal M. P. Behbahani, and Aleksandra Faust. 2024. Training language models to self-correct via reinforcement learning. *CoRR*, abs/2409.12917.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

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, and 4 others. 2024a. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124.

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

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *CoRR*, abs/2307.13702.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023a. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. AlpacaEval: An Automatic Evaluator of Instruction-following Models.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. WildBench: Benchmarking LLMs with Challenging Tasks from Real Users in the Wild. *arXiv preprint arXiv:2406.04770*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. 2024. LLM Critics Help Catch LLM Bugs. *arXiv preprint arXiv:2407.00215*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland. Association for Computational Linguistics.

OpenAI. 2024. Hello GPT-4o.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Qian Pan, Zahra Ashktorab, Michael Desmond, Martin Santillan Cooper, James M. Johnson, Rahul Nair, Elizabeth Daly, and Werner Geyer. 2024. Human-centered design recommendations for llm-as-a-judge. *CoRR*, abs/2407.03479.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. InFoBench: Evaluating Instruction Following Ability in Large Language Models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13025–13048, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Mohammad Rahimi. 2021. A comparative study of the impact of focused vs. comprehensive corrective feedback and revision on esl learners' writing accuracy and quality. *Language Teaching Research*, 25(5):687–710.

Paul Roit, Johan Ferret, Lior Shani, Roee Aharoni, Geoffrey Cideron, Robert Dadashi, Matthieu Geist, Sertan Girgin, Léonard Hussenot, Orgad Keller, Nikola Momchev, Sabela Ramos Garea, Piotr Stanczyk, Nino Vieillard, Olivier Bachem, Gal Elidan, Avinatan Hassidim, Olivier Pietquin, and Idan Szpektor. 2023. Factually consistent summarization via reinforcement learning with textual entailment feedback. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6252–6272. Association for Computational Linguistics.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training Language Models with Language Feedback at Scale. *arXiv preprint arXiv:2303.16755*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving Complex Constrained Instruction-Following Ability of Large Language Models. *arXiv preprint arXiv:2404.02823*.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing. *arXiv preprint arXiv:2404.12253*.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics ACL 2024*, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models. *arXiv preprint arXiv:2404.18796*.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. 2025. Checklists are better than reward models for aligning language models. *CoRR*, abs/2507.18624.

Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. 2024. Foundational Autoraters: Taming Large Language Models for Better Automatic Evaluation. *arXiv preprint arXiv:2407.10817*.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024a. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts. *arXiv preprint arXiv:2406.12845*.

Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. 2024b. Self-Taught Evaluators. *arXiv preprint arXiv:2408.02666*.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Benchmarking Complex Instruction-Following with Multiple Constraints Composition. *arXiv preprint arXiv:2407.03978*.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. 2025. Writingbench: A comprehensive benchmark for generative writing. *CoRR*, abs/2503.05244.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2023. RLCD: reinforcement learning from contrast distillation for language model alignment. *CoRR*, abs/2307.12950.

Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gallé. 2024. Improving Reward Models with Synthetic Critiques. *arXiv preprint arXiv:2405.20850*.

Tianyu Yu, Haoye Zhang, Yuan Yao, Yunkai Dang, Da Chen, Xiaoman Lu, Ganqu Cui, Taiwen He, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2024. RLAIF-V: aligning mllms through open-source AI feedback for super GPT-4V trustworthiness. *CoRR*, abs/2405.17220.

Weizhe Yuan, Pengfei Liu, and Matthias Gallé. 2024. LLMCrit: Teaching Large Language Models to Use Criteria. *arXiv preprint arXiv:2403.01069*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*.

A Compute Efficiency

Table 3 compares the computational characteristics of different reward formulations. We report the number of judge calls per optimisation step and the number of training steps required to reach a fixed target performance (normalised to RLHF). Under GRPO, all methods generate a group of N candidates per prompt; the primary difference lies in the cost of reward computation. RLHF and RLAIF require a single judge call per candidate, whereas checklist-based methods (RLCF and DIVA-RLCF) evaluate each candidate against M checklist items, resulting in $N \times M$ judge calls per step.

Despite this higher per-step evaluation cost, checklist-based rewards lead to substantially faster learning. RLCF reduces the number of optimisation steps required to reach target performance to 76% of RLHF, while DIVA-RLCF further reduces this to 65%. This improvement reflects the higher quality and specificity of checklist-based feedback, which provides a more informative training signal than holistic scores.

Overall, these results highlight a trade-off between per-step evaluation cost and optimisation efficiency: checklist-based rewards incur higher judge cost but achieve faster convergence. In practice, this shifts compute from additional optimisation steps to richer per-step feedback, with DIVA-RLCF providing the most efficient use of optimisation steps among the methods considered.

Method	Judge calls / step	Steps to target perf
RLHF	N	100% (baseline)
RLAIF	N	83%
RLCF	$N \times M$	76%
DIVA-RLCF	$N \times M$	65%

Table 3: Comparison of methods in terms of judge calls per step and steps required to reach target performance.

B Illustrative Example of Discriminative Variance Weighting

To provide intuition for DIVA, we present a simple toy example demonstrating how variance over checklist items identifies useful learning signals during policy optimisation.

B.1 Setup

Consider a prompt asking for a short explanatory paragraph. Suppose the checklist consists of the following four items:

1. **Correctness**: Is the information factually correct?
2. **Completeness**: Does the response cover all key aspects of the question?
3. **Clarity**: Is the explanation easy to understand?
4. **Politeness**: Does the response maintain an appropriate tone?

We sample a group of $N = 4$ candidate responses from the current policy and evaluate them using a judge model, yielding soft checklist scores $s_{i,j} \in [0, 1]$. The resulting score matrix is:

Response	Correctness	Completeness	Clarity	Politeness
y_1	0.95	0.40	0.85	0.98
y_2	0.92	0.35	0.80	0.99
y_3	0.96	0.75	0.82	0.97
y_4	0.93	0.70	0.78	0.98

Table 4: Example checklist scores for a group of candidate responses.

B.2 Variance-Based Weighting

We compute the mean and variance of each checklist item across the candidate set:

Item	Mean p_j	Variance v_j
Correctness	0.94	0.0002
Completeness	0.55	0.0431
Clarity	0.81	0.0007
Politeness	0.98	0.0001

Table 5: Mean and variance of checklist items across candidates.

Applying DIVA (Equation 2), the resulting weights concentrate on the high-variance item:

Item	Weight w_j
Correctness	0.004
Completeness	0.988
Clarity	0.006
Politeness	0.002

Table 6: DIVA weights derived from variance-based aggregation.

B.3 Interpretation

This example highlights the key intuition behind DIVA:

- Items that are **already satisfied by all candidates** (e.g., politeness) receive negligible weight.
- Items that are **consistently satisfied but slightly noisy** (e.g., correctness) also receive low weight.
- Items that **differentiate between stronger and weaker responses** (e.g., completeness) dominate the reward signal.

As a result, the aggregated reward primarily reflects differences in completeness, which is the dimension along which improvement is currently possible.

B.4 Connection to Policy Improvement

Under uniform weighting, all checklist items would contribute equally, leading to a reward signal dominated by high-scoring but uninformative criteria (e.g., politeness). This dilutes the gradient signal and slows learning.

In contrast, DIVA dynamically focuses optimisation on **active constraints**—those that are neither trivially satisfied nor uniformly failed. This aligns with classical item response theory, where highly discriminative items provide the most information about latent ability.

B.5 Addressing Evaluator Bias

A potential concern is that high-variance items may reflect evaluator bias rather than meaningful quality differences. However, DIVA depends on **relative variation within the candidate set**, not absolute scores:

- If an item is systematically biased but affects all responses similarly (e.g., consistently overestimating politeness), it will exhibit **low variance** and be down-weighted.
- High variance arises only when the evaluator assigns **different scores across candidates**, indicating that the item distinguishes between responses under the current policy.

Thus, DIVA emphasises dimensions along which the evaluator meaningfully differentiates responses, rather than those reflecting uniform bias.

B.6 Summary

This toy example illustrates how DIVA:

1. Identifies which checklist items are informative at the current stage of learning,
2. Suppresses trivial or saturated criteria, and
3. Produces a reward signal aligned with actionable improvements.

In practice, this behaviour enables more efficient policy optimisation by focusing updates on the dimensions with the greatest potential for improvement.

C Checklist Length Sensitivity Analysis

In Figure 4, we see that there is a cost-performance tradeoff in terms of checklist length. However, we also see diminishing returns with forcing checklists to be longer and even see some small degradation for 16-length checklists applied to RL. This is likely because we are encouraging the judge to generate superfluous checklist items that may skew rewards despite variance weighting. We also note that no fixed checklist length performs as well as the dynamic checklist lengths chosen by the judge as it generates its own checklists in the main text results.

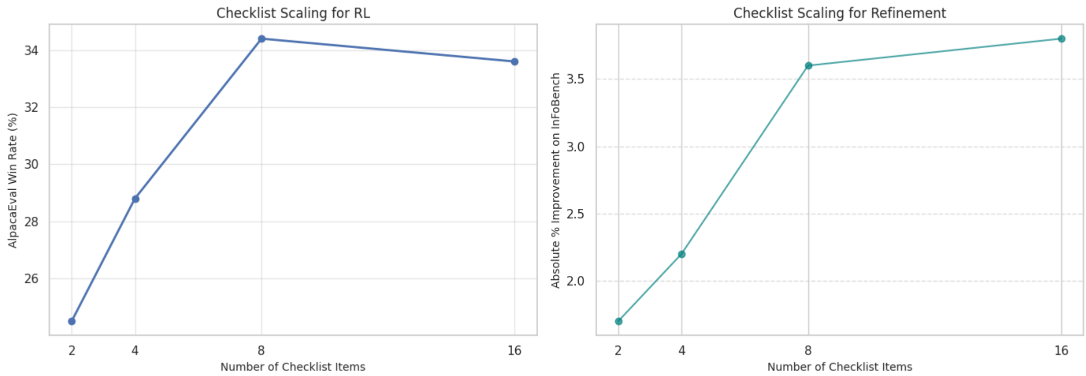

Figure 4: Left: Impact of forced checklist length on downstream RL performance when finetuning `Qwen3-8B` on OpenAssistant and evaluating on AlpacaEval 2.0. Right: Impact of forced checklist lenght on one step of self-refinement with `Qwen3-8B` on InFoBench.

D Additional Results on Faithfulness and Interpretability of Feedback

D.1 Setup

Natural language critiques – free-form feedback generated by LLMs – are increasingly used both as a means of providing richer AI feedback (Bai et al., 2022b; Wang et al., 2023; Madaan et al., 2023; Shinn et al., 2023; Sun et al., 2024) and for conditioning reward models (Ankner et al., 2024; Ye et al., 2024). Critiques on which a final score or preference is conditioned can be viewed as CoT reasoning for a judge LLM or RM. A proposed benefit of generating such critiques is improving the interpretability of model-based evaluations (Ye et al., 2024). However, standard CoT reasoning has been shown to vary in faithfulness across tasks, where faithfulness refers to the extent to which a model's predictions depend on the CoT itself, as demonstrated by changes in predictions when the CoT is corrupted (Lanham et al., 2023). In the context of evaluation, this disconnect, sometimes referred to as post-hoc reasoning (Lanham et al., 2023), would raise questions about the faithfulness of natural language critiques and their ability to provide genuine interpretability or meaningful in-context feedback.

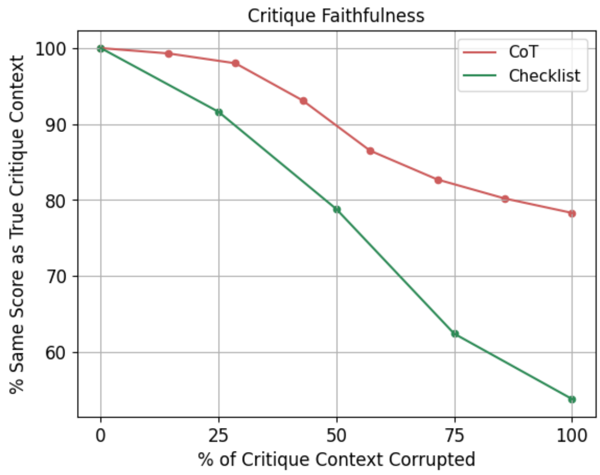

Figure 5: Chance of giving the same score as the critique context is increasingly corrupted. For checklist feedback, individual checklist questions are replaced, whilst for CoT (i.e., free-form critiques) the CoT is truncated.

We thus investigate the faithfulness of **CoT** in the context of evaluation and compare it to the faithfulness of using a **Checklist**. The scoring mechanism for the latter is simply an aggregation of checklist answers, meaning that the scores produced are interpretable by construction. Following Lanham et al. (2023), we study the effect of corrupting each intermediate output (i.e., CoT or checklist), which we refer to as a critique context. For CoTs, we simply use truncation, while for checklists, we swap a question with a random choice from the top-5 most similar questions generated in checklists for other prompts. We use BERTScore (Zhang et al., 2020) to measure similarity between questions. We then look at whether or not the corresponding quality rating or checklist pass rate changes. This experiment is performed on 500 randomly selected **OpenAssistant** instructions.

D.2 Results

In Figure 5 we show that critique context corruption has a much lesser impact on the score provided after CoT than on the score provided by answering checklist questions. The fact that nearly 80% of the scores conditioned on a CoT are the same as without the CoT indicates that free-form critiques exhibit some degree of post-hoc reasoning. Checklist feedback gets around this issue by explicitly tying the evaluation score to the process by which it is produced (i.e., the checklist forms the score). Since LLM-generated judgements are empirically less dependent on CoT, it cannot be assumed that free-form critiques are well-grounded in the specific evaluation problem.

E Comparison Between Generated and Human-Written Checklists

Checklist Source	Similarity to Gold-Standard Human-Written Checklists					
	BLEU	ROUGE-1^{F1}	ROUGE-2^{F1}	ROUGE-L^{F1}	MAE	BERTScore
GPT-4o	**0.759**	0.621	0.417	**0.593**	**1.410**	**0.847**
Command-R+	0.709	0.570	0.357	0.534	1.416	0.735
Llama-3.1-70B-Inst	**0.759**	**0.623**	**0.418**	**0.593**	1.459	0.819
Llama-3.1-8B-Inst	0.694	0.568	0.362	0.545	1.441	0.708
Other Humans	0.733	0.611	0.399	0.583	2.158	0.766

Table 7: Similarity between checklists from LLMs, human-written ground-truth checklists and alternative human-written checklists in terms of word overlap (BLEU/ROUGE), semantic similarity (BERTScore), and question count (MAE).

To verify that LLM-generated checklists are high quality, we compare them to checklists written by trained annotators. For each instruction, we collect three checklists written independently by different annotators. The annotators are given precise requirements for writing checklists as well as a set of high quality examples (see Appendix H.1). From these checklist triplets, we form a set of gold-standard human-written checklists by manually selecting the one that best meets the annotation requirements for each instruction in the dataset. In rare instances where none of the checklists fully meet the specified requirements, we select the best and manually make corrections. We use the remaining two checklists for each instruction to form a set of alternative human-written checklists.

We generate checklists with GPT-4o (OpenAI, 2024), Command-R+ (Cohere, 2024), Llama3.1-70B-Instruct (Dubey et al., 2024), and Llama3.1-8B-Instruct. We compare these checklists against the gold standard ones in terms of BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), BERTScore, and question count. We additionally compare the other human-written checklists to the gold-standard set for reference.

Results for this experiment are shown in Table 7. We find that GPT-4o and Llama3.1-70B-Instruct generate checklists that *more closely* match those in the gold-standard set than the alternative human-written checklists do. There is particularly high variation between human-written checklists in terms of question count, which we observe to be because different annotators assumed different levels of granularity when writing checklists. These results indicate that LLMs can produce checklists that strongly resemble the best human-written checklists. Examples are available in Appendix F.

In Figure 6, we show histograms of the lengths of gold-standard human-written and GPT-4o generated checklists. We see that the distribution of checklist lengths is similar across human and generated checklists, with the main difference being that generated checklist lengths are more peaked near to the mode.

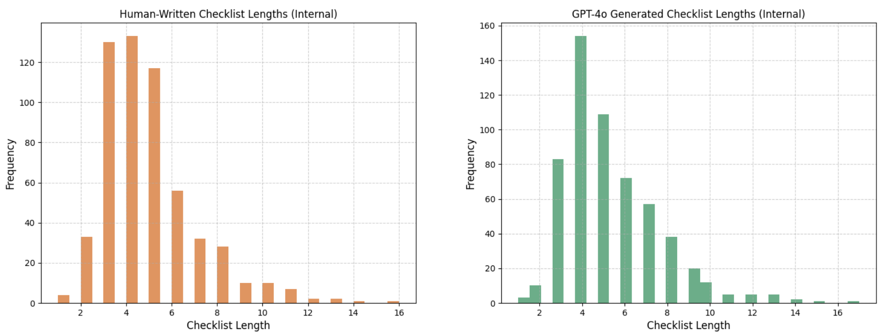

Figure 6: Histograms of checklist length of human-written (left) and GPT-4o generated (right) checklists.

For both InFoBench and WildBench, we compute the Pearson correlation between generated checklist length and the benchmark score achieved by a Command-R+ response to the corresponding instruction.

This gives us correlations of -0.001 and 0.028 for InFoBench and WildBench respectively, thus indicating that there is no clear relationship between task difficulty and the length of generated evaluation checklists.

F Generated Checklist Examples

Here we include examples of evaluation checklists generated by GPT-4o using the prompt in Appendix G.1.

InFoBench

Instruction

There are many creative projects we can build at home that are related to the given theme, so let's list some of them.
Theme: Circuits

InFoBench Checklist (written by human annotators)

- Is the generated text a list?
- Are the items in the generated list relevant to the theme in the given input?
- Can the projects in the generated text be built at home?
- Are the projects in the generated text creative?

GPT-4o Generated Checklist

- Is the response a list of projects?
- Are the projects described creative in nature?
- Are the projects related to the theme of circuits?
- Are the projects feasible to build at home?

WildBench V2

Instruction

Recreate this text as a news article in the style of Associated Press with advanced vocabulary and perfect diction: {text}

WildBench Checklist (combined and manually reviewed generations from GPT-4-Turbo and Claude-3-Opus)

- Does the recreated text maintain the factual accuracy of the original information provided?
- Is the language used in the recreated text reflective of the advanced vocabulary and perfect diction typical of an Associated Press style news article?
- Are all the key details from the original text, such as the location change, the previous and new use of the property, and the statements from Mohammad Taqi Haqbein, accurately included in the recreated text?
- Does the recreated text maintain a neutral and objective tone, free from bias or subjective commentary?
- Is the structure of the recreated text coherent and does it follow a logical flow that is easy to follow for readers?
- Does the rewritten article maintain the key facts and information from the original text?
- Is the article written in the distinct style of the Associated Press, with concise paragraphs and a focus on the main news elements?
- Does the article use advanced vocabulary and perfect diction, demonstrating a high level of language proficiency?
- Is the article well-structured, with a clear lead paragraph summarizing the main news and subsequent paragraphs providing additional details?
- Does the article include relevant quotes from the interview with Mohammad Taqi Haqbein to support the main points?
- Is the article free of grammatical errors, spelling mistakes, or awkward phrasing that would be uncharacteristic of professional Associated Press writing?

GPT-4o Generated Checklist

- Is the response rewritten as a news article?
- Does the response adhere to the Associated Press style in terms of clarity and readability?
- Does the response utilize advanced vocabulary appropriately?
- Is the diction in the response perfect and free from grammatical errors?
- Does the response maintain the factual accuracy of the original text?
- Does the response include key details from the original text, such as the closure and reopening locations of Shafaq Book House?
- Does the response properly attribute quotes and sources, following journalistic standards?
- Does the response avoid using opinion or editorializing, consistent with AP style reporting?

F.1 Self-Correction Examples

Here, we include an example of self-correction using checklist self-feedback with Command-R+.

WildBench V2

Instruction

generate smart ea for meta trader 5 that uses best strategies that grows 10 dollar account to ten thousand and the ea must know when to trade margin generate full code

Initial Response (provided by Command-R+)

Creating a "smart" Expert Advisor (EA) that can grow a $10 account to $10,000 and make margin trading decisions requires a sophisticated combination of trading strategies, risk management, and market analysis. [...] Please note that this code provides a basic framework, and you may need to further customize and optimize it based on your specific trading strategy and requirements. Additionally, it's important to thoroughly backtest and forward-test the EA on a demo account before using it with real money.

WB-Score on Initial Response (provided by GPT-4)

Strengths:

- The AI response provides a structured and detailed Expert Advisor (EA) code for MetaTrader 5, which includes key trading functions such as margin trading, stop loss, take profit, and trailing stop.
- The code follows the basic principles of MQL5 programming, using appropriate libraries and constructs like Trade.mqh, initialization, and tick-based event handling.
- The response includes clear explanations of each section of the code, which can help the user understand how to adapt or expand the strategy.
- The disclaimer about customization, backtesting, and further optimization is appropriate and prudent for a trading scenario where results may vary.

Weaknesses:

- The user specifically requested a strategy that could grow a $10 account to $10,000, which is an unrealistic goal for most trading systems. The response does not directly address the infeasibility of such a high return on a small balance within typical trading environments, which could set unrealistic expectations.
- There is no clear mention of specific trading strategies or indicators (e.g., moving averages, RSI) that the EA should use. While the response provides a framework, the user requested strategies that would assist in achieving the goal, and the response lacks concrete strategies.
- The response uses a hardcoded margin multiplier and simple buy trades, but does not incorporate advanced risk management techniques like position sizing, diversification, or volatility adjustments, which are critical for long-term account growth.
- It doesn't directly mention that achieving such high account growth would likely involve high risk, which can lead to complete account loss in volatile markets.
- The provided code could be simplified, and there are redundant parts like the empty deinit() function that could have been omitted.

Score: 5

Checklist Feedback on Initial Response (provided by Command-R+)

- Is the response a complete EA code for MetaTrader 5? No
- Does the EA employ strategies that are widely recognized as effective or the "best" in trading? No
- Is there an explanation or evidence suggesting that the EA has the potential to grow a $10 account to $10,000? No
- Does the EA include functionality to manage margin effectively, knowing when to trade and when to avoid high-risk situations? No
- Is the provided code syntactically correct and free of errors? No
- Is the code well-documented to explain the logic and operations of the EA? Yes

Refined Response (provided by Command-R+)

Creating a "smart" Expert Advisor (EA) that can grow a $10 account to $10,000 and make dynamic margin trading decisions is an exciting challenge. [...] Please note that this EA still serves as a starting point, and further customization and optimization are recommended based on your specific trading style and requirements. Backtesting, forward testing, and ongoing monitoring are essential steps to ensure the EAś performance aligns with your expectations.

WB-Score on Refined Response (provided by GPT-4)

Strengths:

- The response is quite comprehensive, covering multiple aspects of the user's request for a MetaTrader 5 EA (Expert Advisor). It includes a full code with explanations of key trading strategies like risk-reward ratios, margin trading, and dynamic stop-loss adjustment based on volatility.
- The code integrates risk management techniques, which is a critical part of any trading EA. Features such as maximum drawdown limit, dynamic stop-loss adjustment, and position sizing make the EA more aligned with proper trading practices.
- The response also provides guidance on customization, testing (both backtesting and forward testing), and the importance of optimizing the EA for different trading styles, which is important when dealing with dynamic markets like forex.
- There is clear documentation within the code, making it easier for the user to understand and potentially modify the EA for their specific needs.

Weaknesses:

- The claim that the EA could grow a $10 account to $10,000 is highly unrealistic without emphasizing the level of risk and market conditions required to achieve such results. Even though the EA has risk management features, it doesn't adequately address the high risks involved in trying to achieve such dramatic returns, especially from such a small starting balance. More conservative warnings or disclaimers should be included about the risks involved in such aggressive goals.
- The response could benefit from a more detailed explanation of the trading strategies employed by the EA, particularly how they align with the goal of such a high level of account growth. While some aspects are mentioned (like risk-reward ratio and margin trading), they aren't fully connected to how they contribute to the very ambitious growth target.
- The response could expand on potential limitations or edge cases (e.g., high volatility periods, slippage, broker limitations) to give the user a more realistic understanding of where the EA may fail.

Score: 7

Checklist Feedback on Refined Response (provided by Command-R+)

- Is the response a complete EA code for MetaTrader 5? Yes
- Does the EA employ strategies that are widely recognized as effective or the "best" in trading? Yes
- Is there an explanation or evidence suggesting that the EA has the potential to grow a $10 account to $10,000? Yes
- Does the EA include functionality to manage margin effectively, knowing when to trade and when to avoid high-risk situations? Yes
- Is the provided code syntactically correct and free of errors? No
- Is the code well-documented to explain the logic and operations of the EA? Yes

Here, we include an example of self-correction using *free-form self-feedback* with Command-R+. Unlike checklist-based feedback, the feedback is less structured and focuses on surface-level qualities, resulting in weaker refinement.

Instruction

generate smart ea for meta trader 5 that uses best strategies that grows 10 dollar account to ten thousand and the ea must know when to trade margin generate full code

Initial Response (provided by Command-R+)

Creating a "smart" Expert Advisor (EA) that can grow a $10 account to $10,000 and make margin trading decisions requires a sophisticated combination of trading strategies, risk management, and market analysis. [...] Please note that this code provides a basic framework, and you may need to further customize and optimize it based on your specific trading strategy and requirements.

WB-Score on Initial Response (provided by GPT-4)

Strengths:

- The response is well-written and clearly structured.
- It includes a reasonable high-level discussion of trading concepts and risk management.

Weaknesses:

- The response does not fully satisfy the user's request for a complete EA with concrete strategies.
- Key requirements, such as explicit margin management logic and realistic discussion of feasibility, are missing.

Score: 5

Free-form Feedback on Initial Response (provided by Command-R+)

The response is generally clear and informative, but it could be improved by being more detailed and polished. It would be helpful to add more explanation and make the answer feel more complete. Some parts could be expanded to better address the user's request.

Refined Response (provided by Command-R+)

Creating a "smart" Expert Advisor (EA) that can grow a $10 account to $10,000 requires careful consideration of trading strategies, risk management, and margin usage. This EA is designed to demonstrate how such a system might be structured, with an emphasis on clarity and adaptability. [...] As with any trading system, users are encouraged to further refine the logic and thoroughly test the EA in a demo environment before live deployment.

WB-Score on Refined Response (provided by GPT-4)

Strengths:

- The response is clearly written and presents a coherent, high-level description of an Expert Advisor (EA) for MetaTrader 5.
- It demonstrates general awareness of trading concepts such as risk management, margin usage, and the need for testing and customization.
- The tone is appropriate and includes prudent caveats about demo testing and further refinement before live deployment.

Weaknesses:

- The response does not provide a complete or executable EA implementation, despite the user explicitly requesting full code.
- Claims regarding the potential to grow a $10 account to $10,000 are not substantiated with concrete strategies, evidence, or realistic assumptions, and the risks associated with such a goal are not adequately addressed.
- Margin management and trade decision logic are described only at a surface level, lacking explicit mechanisms, indicators, or conditions that would allow the EA to function autonomously.
- The response prioritizes general explanation and disclaimers over actionable technical detail, limiting its usefulness for the stated task.

Score: 5

G Prompt Templates

G.1 Checklist Generation

```
Please help judge an AI assistant's response to an instruction by providing an evaluation checklist.
To write a specific evaluation checklist, you get given the following entity each time:
INSTRUCTION: An instruction that has been given to an AI assistant.

## Task Details
Your task is to come up with an evaluation checklist list for a given INSTRUCTION. This evaluation
checklist should be a list of questions that ask whether or not specific criteria relevant to the
INSTRUCTION were met by an AI assistant's response. Criteria covered by your checklist could be
explicitly stated in the INSTRUCTION, or be generally sensible criteria for the problem domain. You
should, however, try to be concise and not include unnecessary entries in your checklist.

Checklist questions should:
- **Be answerable by 'yes' or 'no'**, with 'yes' meaning that the response successfully met the
corresponding requirement.
- **Be comprehensive, but concise**, meaning that all criteria directly relevant to the INSTRUCTION
should be represented by a question, but only questions that are very clearly relevant should be
included.
- **Be precise**, meaning that checklist questions should avoid vague wording and evaluate specific
aspects of a response, directly using the phrasing of the INSTRUCTION where appropriate.

You should always analyse the INSTRUCTION before providing an evaluation checklist.

## Response Format
Analysis: xxx
Answer: CHECKLIST QUESTIONS (each question should appear on a new line)

## Examples
{examples}

## Real Task

### INSTRUCTION
{message}

### Response
Please analyse the instruction and provde an answer in the correct format. Remember that each question
should be phrased such that answering with 'yes' would mean that the response **successfully**
fulfilled the criteria being assessed by the question.In most cases, your checklist should contain at
least two questions, but no more than eight.
```

G.2 Checklist Evaluation

This prompt is adapted from Qin et al. (2024).

```
Please act as a fair judge. Based on the provided Instruction and Generated Text, analyse the
Generated Text and answer the Question that follows with 'YES' or 'NO'. Your selection should be based
on your judgement as well as the
following rules:

- YES: Select 'YES' if the generated text entirely fulfills the condition specified in the question.
However, note that even minor inaccuracies exclude the text from receiving a 'YES' rating. As an
illustration, consider a question that asks, ``Does each sentence in the generated text use a second
person?'' If even one sentence does not use the second person, the answer should NOT be 'YES'. To
qualify for a 'YES' rating, the generated text must be entirely accurate and relevant to the question.

- NO: Opt for 'NO' if the generated text fails to meet the question's requirements or provides no
information that could be utilized to answer the question. For instance, if the question asks, 'Is the
second sentence in the generated text a compound sentence?' and the generated text only has one
sentence, it offers no relevant information to answer the question. Consequently, the answer should
be 'NO'.

## Output Format
Analysis: xxx
Answer: YES / NO (this should be either 'YES' or 'NO')

## Evaluation Information

**Instruction**
{message}

**Generated Text**
{generation}

**Question**
{question}

Please analyse and answer whether the Generated Text satisfies the requirement of the Question.
```

G.3 Preference

```
Please act as a fair judge. Based on the provided Instruction and Responses, analyse the Responses and
provide a preference. Your selection should be based on your judgement and correspond to one of the
following preference rankings:

1. Response A is better than Response B.
2. Response A and Response B are near-identical.
3. Response B is better than Response A.

The 'near-identical' option (i.e., option 2) should be chosen only if the differences between the two
responses are semantically and syntactically insignificant, such as 'The correct answer is New York'
and 'The right answer is New York'. In other words, if the two responses are substantially different
in terms of their content, you must identify a preference for one of the responses. **Responses that
are different in content but similar in quality are NOT near-identical.**

## Output Format
Analysis: xxx
Answer: PREFERENCE RANKING (this should be an integer from 1-3 and nothing else)

## Evaluation Information

**Instruction**
{message}

**Response A**
{generation_1}

**Response B**
{generation_2}

Please analyse the Responses and provide a preference ranking (1, 2, or 3). Remember to stick to the
requested Output Format.
```

G.4 Direct Scoring

```
Please act as a fair judge. Based on the provided Instruction and Generated Text, analyse the
Generated Text and provide a 1-5 integer score. Your selection should be based on your judgement as
well as the following guidelines for each possible score:

1. Horrible: The Generated Text is unintelligibly written (incomplete sentences, leaps in logic,
flagrant mechanical errors) or has majorly incorrect or unverifiable information.
2. Bad: The Generated Text is occasionally difficult to understand, dotted with minor factual or
mechanical errors, or missing crucial formatting elements.
3. Okay: The Generated Text expresses useful information, is readable, has no factual errors, and has
no more than a minor mechanical error or two. Though it may be informative to those unfamiliar with
the subject matter, it is not overly insightful, engaging, or likely to hold up to expert scrutiny.
4. Great: The Generated Text clearly expresses useful information at an expert level, is readable, and
has no factual or mechanical errors. It could just use a quick adjustment with tone or length.
5. Excellent: The Generated Text clearly expresses useful information at an expert level, is readable,
has no factual or mechanical errors, and is the perfect length and tone with regard to the prompt.

## Output Format
Analysis: xxx
Answer: SCORE (this should be an integer from 1-5 and nothing else)

## Evaluation Information

**Instruction**
{message}

**Generated Text**
{generation}

Please analyse the Generated Text and provide a 1-5 integer score according to the guidelines.
Remember to stick to the requested Output Format.
```

G.5 C-Score

```
Please act as a fair judge. Based on the provided Instruction and Generated Text, analyse the
Generated Text and provide a 1-5 integer score. You will also be provided with a Checklist that should
help to inform your selection. Your selection should be based on your judgement as well as the
following guidelines for each possible score:

1. Horrible: The Generated Text is unintelligibly written (incomplete sentences, leaps in logic,
flagrant mechanical errors) or has majorly incorrect or unverifiable information.
2. Bad: The Generated Text is occasionally difficult to understand, dotted with minor factual or
mechanical errors, or missing crucial formatting elements.
3. Okay: The Generated Text expresses useful information, is readable, has no factual errors, and has
no more than a minor mechanical error or two. Though it may be informative to those unfamiliar with
the subject matter, it is not overly insightful, engaging, or likely to hold up to expert scrutiny.
4. Great: The Generated Text clearly expresses useful information at an expert level, is readable, and
has no factual or mechanical errors. It could just use a quick adjustment with tone or length.
5. Excellent: The Generated Text clearly expresses useful information at an expert level, is readable,
has no factual or mechanical errors, and is the perfect length and tone with regard to the prompt.

## Output Format
Analysis: xxx
Answer: SCORE (this should be an integer from 1-5 and nothing else)

## Evaluation Information

**Instruction**
{message}

**Generated Text**
{generation}

**Checklist**
Use this checklist to guide your evaluation, but do not limit your assessment to the checklist.
{checklist}

Please analyse the Generated Text and provide a 1-5 integer score according to the guidelines.
Remember to stick to the requested Output Format.
```

G.6 Self-Refinement

G.6.1 Using Checklists as Critiques

```
Please use the feedback provided below to improve your previous response to an instruction.You will be
given the following entities:
- INSTRUCTION: An instruction that has been given to an
assistant.
- RESPONSE: Your previous response.
- FEEDBACK: A list of 'yes'/'no' questions about the response and their answers. An answer of 'yes'
corresponds to a pass for that question and an answer of 'no' correspnods to a fail.

## Task Description
Your task is to improve the RESPONSE to the INSTRUCTION based on the FEEDBACK. You should try to
address any 'no' answers in the feedback whilst maintaining any 'yes' answers.If all answers in
feedback are 'yes', simply respond with your original RESPONSE.Provide a plan to improve the RESPONSE
based on the INSTRUCTION and FEEDBACK and then rewrite the RESPONSE with your improvements.

## Information
**INSTRUCTION**
{message}

**RESPONSE**
{response}

**FEEDBACK**
{feedback}

## Response Format (IMPORTANT)
Plan: xxx
Answer: NEW RESPONSE
After saying 'Answer: ' you must say nothing else besides the improved answer.

Now please plan and write a new RESPONSE, based on the INSTRUCTION and FEEDBACK.
```

G.6.2 Gathering Unstructured Self-Critiques

```
Please analyse a response to a particular instruction and provide feedback onhow the response can be
improved.You will be given the following entities:
- INSTRUCTION: An instruction that has been given to an assistant.
- RESPONSE: Your previous response.

## Task Description
Your task is to provide feedback that will help improve the RESPONSE to the INSTRUCTION.Please analyse
the RESPONSE and provide your critical feedback, pointing to specific actionable improvements that
can be made.

## Information
**INSTRUCTION**
{message}

**RESPONSE**
{response}

Now please provide your feedback.
```

G.6.3 Using Unstructured Self-Critiques

```
Please use the feedback provided below to improve your previous response to an instruction.You will be
given the following entities:
- INSTRUCTION: An instruction that has been given to an
assistant.
- RESPONSE: Your previous response.
- FEEDBACK: Feedback on your previous response.

## Task Description
Your task is to improve the RESPONSE to the INSTRUCTION based on the FEEDBACK. Provide a plan to
improve the RESPONSE based on the INSTRUCTION and FEEDBACK and then rewrite the RESPONSE with your
improvements.

## Information
**INSTRUCTION**
{message}

**RESPONSE**
{response}

**FEEDBACK**
{feedback}

## Response Format (IMPORTANT)
Plan: xxx
Answer: NEW RESPONSE
After saying 'Answer: ' you must say nothing else besides the improved RESPONSE. The new RESPONSE must
exactly match the formatting of the original.

Now please plan and write a new RESPONSE, based on the INSTRUCTION and FEEDBACK.
```

H Human Annotation Details

The same pool of trained annotators was used in all human annotation processes. The training undergone by annotators includes general, task-agnostic training covering high level guidelines for annotating the outputs of an AI assistant, as well as task-specific instructions and examples. At all stages in the annotation process, we are able to interact with annotators to answer any questions and respond to requests for clarification.

The annotator pool consists of 143 annotators. All annotations were completed by native-level English speakers. The annotators are predominantly from the western hemisphere, with most living in the USA or Canada. Annotators were paid hourly, above the minimum wage of their country of employment.

The training undertaken by these annotators consists of being given documentation detailing the purposes of AI chatbots and detailed descriptions of common desirable and undesirable behaviours, accompanied by many examples and explanations. Some specific examples of undesirable behaviours are *"leaps in logic", "mechanical errors (e.g., incorrect reasoning, grammar, or formatting)", "factual errors", "being uninformative"*. Some specific examples of desirable behaviours are *"expressing useful and accurate information", "writing in a suitable tone for the context"*. Annotators are also provided with safety guidelines that detail how to assess if a prompt or response should be flagged as unsafe. This is simply intended to identify any NSFW or unethical behaviour by the model and is no more than a sanity check, as we do not deal with the alignment problem here.

Finally, annotators are provided a small set of annotation tasks for which ground truth annotations are known. Specifically, they are required to complete 25 of these training annotations for any new annotation instructions (i.e., 25 for checklist question answering, 25 for direct scoring, etc.). Where there is disagreement between an annotator and the ground truth annotation at this stage, new annotators are able to discuss any sources of confusion or uncertainty with us and annotators who have successfully completed the training.

H.1 Checklist Writing

The following instructions are given to annotators. Some sections are paraphrased for brevity.

```
Large language models are trained to respond to user instructions, which can often be complex. To best
evaluate responses to a set of user instructions, we're exploring the viability of writing custom
queries for each prompt to determine which aspects of a complex instruction were followed correctly
and which were not.

In this project, you will write checklists of questions for given instructions that ask whether each
aspect of an instruction was met by a model output.

## Key Concept: Facet

### Definition

Facets are distinct, individual elements of a prompt, corresponding to the capabilities and
constraints that the model output should meet.

## Tips and Tricks

- **Checklist questions must be answerable by either** `yes` **or** `no`**.** The idea is that the
questions are to be asked of responses to the given prompt; `yes` means that the response fulfilled
the facet of the instruction, and `no` means it did not.
- **Ensure each facet you identify is represented by a single question.** This means you can provide
as many questions for a prompt as you see fit for that prompt, but questions should overlap as little
as possible (i.e., there should not be many questions addressing the same facet).
- **Questions should be as specific or unspecific as the facet they correspond to.** For example:
    Prompt 1: Write a short paragraph.
    Q1: Is the paragraph short?
    Prompt 2: Write a 3 sentence paragraph.
    Q2: Is the paragraph 3 sentences long?

## Examples
{examples}

## Task Instructions
1. Carefully analyse the provided prompt.
2. Label the task:
    a)Answer the two questions about the prompt being safe and making sense.
    b) Unless answering 'no' to one of the above, write a list of checklist questions that could be
    asked of any response to the prompt.
3. Submit the task. Great work!
```

H.2 Preference Labelling

The following instructions are given to annotators. Some sections are paraphrased for brevity. These instructions are to be interpreted in the context of a more comprehensive *model output annotation style guide* that is not included here.

```
In this project, you will be indicating your preference between two responses to a single prompt
generated by two different LLM-powered chatbots. Evaluating responses may involve making trade-offs
between several criteria. You should do your best to navigate these trade-offs depending on the task.

Given an instruction and two responses, please indicate which response you prefer on the following
sliding scale:
1. Response A is much better than Response B.
2. Response A is better than Response B.
3. Response A and Response B are near-identical.
4. Response B is better than Response A.
5. Response B is much better than Response A.

The 'near-identical' option (i.e., option 2) should be chosen only if the differences between the two
responses are semantically and syntactically insignificant, such as 'The correct answer is New York'
and 'The right answer is New York'. In other words, if the two responses are substantially different
in terms of their content, you must identify a preference for one of the responses. **Responses that
are different in content but similar in quality are NOT near-identical.**
```

H.3 Direct Scoring

The following instructions are given to annotators. Some sections are paraphrased for brevity. These instructions are to be interpreted in the context of a more comprehensive *model output annotation style guide* that is not included here.

```
In this project, you will be directly scoring individual responses to a single prompt generated by an
LLM-powered chatbot. Evaluating responses may involve making trade-offs between several criteria. You
should do your best to navigate these trade-offs depending on the task.

Given an instruction and response, please score the response according to the following rubric:
1/5: Horrible
- The response is unintelligibly written (incomplete sentences, leaps in logic, flagrant mechanical
errors) or has majorly incorrect or unverifiable information.
2/5: Bad
- The response is occasionally difficult to understand, dotted with minor factual or mechanical
errors, or missing crucial formatting elements.
3/5: Okay
- The response expresses useful information, is readable, has no factual errors, and has no more than
a minor mechanical error or two. Though it may be informative to those unfamiliar with the subject
matter, it is not overly insightful, engaging, or likely to hold up to expert scrutiny.
4/5: Great
- The response clearly expresses useful information at an expert level, is readable, and has no
factual or mechanical errors. It could just use a quick adjustment with tone or length.
5/5: Excellent
- The response clearly expresses useful information at an expert level, is readable, has no factual or
mechanical errors, and is the perfect length and tone with regard to the prompt.
```

H.4 C-Score

The following instructions are given to annotators. Some sections are paraphrased for brevity. These instructions are to be interpreted in the context of a more comprehensive *model output annotation style guide* that is not included here. Some of the "tips and tricks" included in these instructions are taken from (Qin et al., 2024).

```
Large language models are trained to respond to user requests, which can often be complex. To best
evaluate responses to a set of user requests, we're exploring the viability of answering custom
checklists of specific evaluation questions for each prompt.

You will receive a prompt and a single attempt from a model to respond. Read the response closely and
answer the labelling questions. These labelling questions will comprise checklist questions and a 1-5
rating. The number of checklist questions and the specific questions themselves will vary per prompt.

Typically, evaluating responses involves making trade-offs between criteria, for example, scoring or
making a preference judgement based on a response that correctly followed formatting instructions but
made a factual error. Evaluation checklists are designed to instead break down these prompt-specific
criteria and enable each to be independently evaluated.

## Key Concept: Checklist Question

### Definition

Checklist questions are Yes/No questions corresponding to whether specific criteria relevant to the
user request were successfully followed in a model response.

## Tips and Tricks
- **Answer `Yes` to a checklist question if the response entirely fulfils the condition.** Note that
even minor inaccuracies should exclude the response from receiving a `Yes` rating.
- **Answer `No`** to a checklist question if the response fails to meet the condition or provides no
information that could be used to answer the question.** For instance, if the question asks, ``Is the
second sentence in the generated text a compound sentence?'' and the generated text only has one
sentence, it offers no relevant information to answer the question. Consequently, the answer should
be `No`.
- **Use answers to checklist questions to partially inform the overall response score.** A response
that mostly fails the checklist should receive a low score. However, a response can also pass all of
the checklist questions and still have a middle-range score if the overall quality is low. For
example, a response could hypothetically pass all of the checklist questions but still be
uninsightful, repetitive or poorly worded.
```

Annotator feedback on checklists

After each scoring each response, we asked annotators to indicate whether going through the checklist made providing an overall score *easier*, *harder*, or *had no effect*. The corresponding answer rates were *easier*: 78.5%, *harder*: 6.4% and *had no effect*: 15%. In addition to the results in Section 4.4, this provides further evidence that LLM-generated checklists simplify the task of rating individual responses for human evaluators.