Are Checklists Really Useful for Automatic Evaluation of Generative Tasks?

Momoka Furuhashi[1,2] **Kouta Nakayama**[2] **Takashi Kodama**[2] **Saku Sugawara**[3,2]
[1]Tohoku University [2]Research and Development Center for Large Language Models, National Institute of Informatics [3]National Institute of Informatics
furuhashi.momoka.p4@dc.tohoku.ac.jp {nakayama,tkodama,saku}@nii.ac.jp

Abstract

Automatic evaluation of generative tasks using large language models faces challenges due to ambiguous criteria. Although automatic checklist generation is a potentially promising approach, its usefulness remains underexplored. We investigate whether checklists should be used for all questions or selectively, generate them using six methods, evaluate their effectiveness across eight model sizes, and identify checklist items that correlate with human evaluations. Through experiments on pairwise comparison and direct scoring tasks, we find that selective checklist use tends to improve evaluation performance in pairwise settings, while its benefits are less consistent in direct scoring. Our analysis also shows that even checklist items with low correlation to human scores often reflect human-written criteria, indicating potential inconsistencies in human evaluation. These findings highlight the need to more clearly define objective evaluation criteria to guide both human and automatic evaluations. [1]

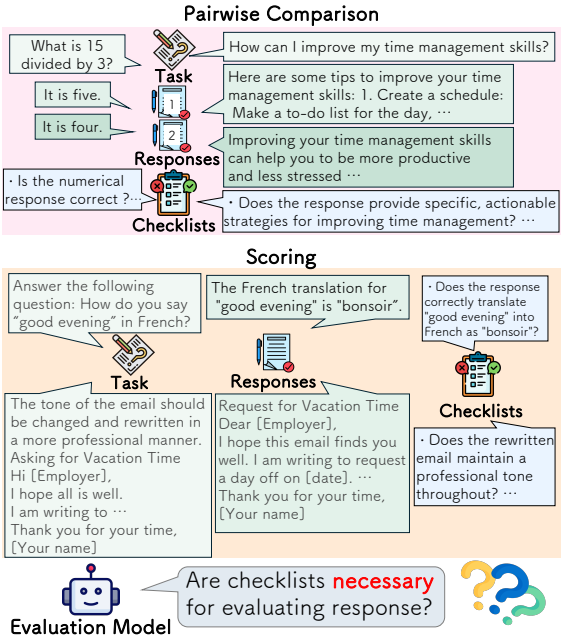

Figure 1: Examples of using checklists in automatic evaluation by LLMs. Existing studies use checklists even in situations where fine-grained criteria may be unnecessary for evaluating the responses.

1 Introduction

Automatic evaluation using large language models (LLMs) has been widely adopted for generative tasks (Chang et al., 2024; Ferraz et al., 2024; Gu et al., 2025; Li et al., 2025). This approach provides an efficient and scalable alternative to costly and time-consuming human evaluation. However, it faces two major challenges. First, establishing clear and consistent evaluation criteria remains difficult, leading to potential ambiguity in scoring. Second, the correlation between LLM-based automatic evaluation and human evaluation is often unstable, limiting its reliability.

To address these challenges, previous studies have introduced a *checklist* approach that decomposes evaluation criteria into specific, fine-grained items (Lee et al., 2024; Qin et al., 2024; Lin et al., 2025). As shown in Figure 1, when an LLM evaluates responses to mathematical problems, evaluator models can refer to a detailed checklist with items such as "Does the calculated value match the correct answer?" and "Does the response contain unnecessary decimal points?" Although checklists are easy to use and understand, previous studies have not fully investigated three key aspects: when checklists are necessary, how we can create them, and how checklist items relate to alignment with human evaluation. We examine the usefulness of checklists in automatic evaluation by answering the following three questions shown in Figure 2. RQ1: *Can we determine whether a checklist is necessary for LLM evaluators?* RQ2: *How can we create useful checklists?* RQ3: *Which checklist items con-*

[1]Our code is available at https://github.com/momo0817/checklist-effectiveness-study

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 10630–10653
November 4-9, 2025

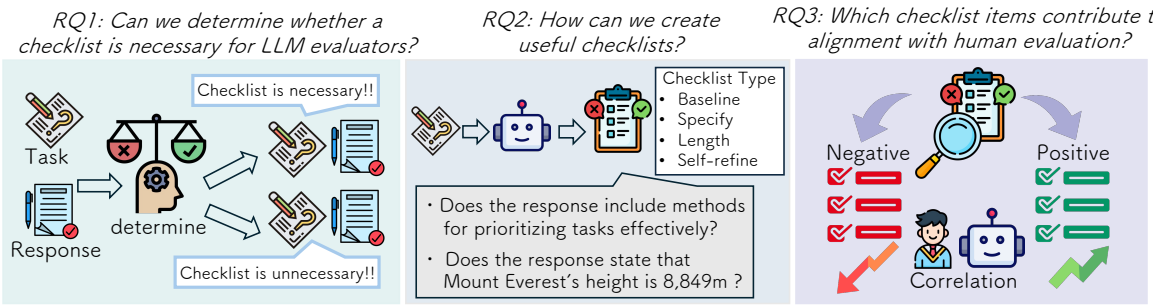

Figure 2: Our research questions. First, we investigate whether we can identify which responses require checklist evaluation (RQ1). Next, we study how checklist generation affects alignment with human evaluations by evaluating eight models of different sizes (RQ2), comparing six different generation methods. Finally, we analyze which checklist items contribute most to alignment with human evaluation (RQ3).

tribute to alignment with human evaluation?

To investigate RQ1 and RQ2, we conduct three controlled experiments. First, focusing on the consistency of multiple automatic evaluations, we set a threshold to decide whether a checklist is necessary for each evaluation case. Second, we investigate which checklist features improve correlation. In this process, we create three methods (control checklist length, more specified generation, and self-refine) in addition to the baseline and existing method. Third, we evaluate a total of eight models, including gpt-4o-2024-08-06 (OpenAI, 2024) and Qwen2.5-7B-Instruct (Yang et al., 2025), to investigate which checklist items contribute to alignment with human evaluation. For each of these three aspects, we test their effect on alignment using two types of human evaluation data: pairwise comparison and direct scoring. To investigate RQ3, we also conduct a more detailed checklist-based analysis on the smallest model. We analyze checklist items both quantitatively and qualitatively, focusing on their overlap with human-written ones.

Our experiments yield three key findings: First, we observe that the effectiveness of selective checklist application varies by task; in some cases, it achieves comparable or better correlations than full application, while in others it does not. This suggests that checklist use is not universally beneficial. Second, our analysis reveals that the most useful checklist creation method varies across different evaluation models and tasks, suggesting that no single approach works best in all settings. Third, our analysis shows that many checklist items, although useless for improving human correlation, still overlap substantially with human-written items. This suggests that inconsistencies may stem from the subjective nature of human evaluations and underscores the need to rethink the objective criteria we expect from responses.

Our contributions are summarized as follows:

- We find that selective checklist use sometimes improves evaluation outcomes, suggesting that omitting checklists can be justified in specific settings.
- We show that no universally optimal checklist generation method exists, as usefulness varies significantly depending on the evaluation model and use case.
- We find that even checklist items with low correlation to human evaluations often overlap with human-written ones, indicating they may still capture valid criteria. This highlights the subjective nature of human evaluations and calls for more objective evaluation design.

2 Related Work

Recent studies have investigated the use of LLMs as evaluators for generative tasks (Chang et al., 2024; Gu et al., 2025; Li et al., 2025). Automatic evaluation methods using LLMs fall into two methods: pairwise comparison (Wei et al., 2022; Wang et al., 2024; Zeng et al., 2024; Lambert et al., 2025; Tan et al., 2025) and direct scoring (Ye et al., 2024; Kim et al., 2024; Liu et al., 2024). These approaches rely on human evaluation as gold labels and assess performance via correlation and agreement rates. However, both methods have inherent limitations: pairwise comparison suffers from ambiguity in evaluation criteria, while direct scoring faces difficulties in metric definition.

Fine-grained Evaluation Criteria Previous studies have explored breaking down evaluation

criteria into smaller components to improve correlation. Min et al. (2023) evaluate factual accuracy by splitting responses into individual statements, each containing a single piece of information. Kim et al. (2024) manually create 50 scoring rubrics focusing on critical aspects of response evaluation and then expand these rubrics using GPT-4. Ye et al. (2024) enhances evaluation reliability by decomposing their evaluation into skill-level scoring sets for each instruction.

Checklist-based Evaluation Previous studies have proposed a checklist-based approach that breaks down complex evaluation criteria into smaller, more specific points of evaluation (Lee et al., 2024; Qin et al., 2024; Lin et al., 2025; Cook et al., 2024). CheckEval (Lee et al., 2024) decomposes evaluation criteria such as fluency into manually created checklists for summarization tasks, where each item requires a binary yes/no response, with the final score derived from the ratio of *yes* responses. Additionally, Qin et al. (2024) manually create 2,500 checklists based on 500 distinct instructions and conduct comprehensive evaluations using six evaluation models. Furthermore, WildBench (Lin et al., 2025) establishes a benchmark for evaluating LLMs on real-world-inspired tasks, generating five to ten checklist items for each question task by using GPT-4-Turbo and Claude-3-Opus. While Cook et al. (2024) shows that LLM-generated checklists improve correlations, previous studies have not investigated when checklists are actually needed or how useful they are.

3 Dissecting Checklist-based Evaluation

To investigate RQ1 and RQ2, we conduct three controlled experiments: First, we investigate whether it is possible to identify instances where checklists are unnecessary for automated evaluation (Session 3.1); second, we evaluate six different checklist generation methods to determine which types of checklists are most useful (Session 3.2); third, we examine the usefulness of checklists using eight different models, ranging from small to large size, to assess their practicality (Session 3.3).

3.1 Identifying When Responses Need Checklist Evaluation

To address RQ1: *Can we determine whether a checklist is necessary for LLM evaluators?*, we compare how well model evaluations correlate with human evaluation both with and without checklists. We hypothesise that checklists are necessary when LLM evaluations lack consistency. Therefore, we conduct multiple evaluations without checklists and apply checklists only to responses that receive inconsistent labels above a threshold. We then compare this selective approach against two baselines: using no checklists at all (*None*) and using checklists for every response (*All*). Through this comparison, we determine if targeting the checklist use to low-reliability cases improves overall correlation. We conduct this analysis across different checklist variations described in Section 3.2.

3.2 Checklist Generation Policy

To address RQ2: *How can we create useful checklists?*, we vary the level of detail and number of items. To better control these factors, we generate checklists for evaluating generative tasks using three methods. We analyze how each method correlates to identify the most useful checklist types. Below, we describe each checklist generation method.

Baseline In this study, we examine how limiting the number of items and adjusting the level of detail affect checklist generation. We incorporate the following three elements: (1) Each item must allow a simple *yes* or *no* answer, where *yes* confirms success. (2) Criteria must directly relate to essential task requirements. (3) Questions must use specific wording and reference input phrasing directly, concrete wording that directly relates to the task, avoiding vague or ambiguous language.

Specify Previous studies distinguish between two types of checklist items: surface-level evaluation (e.g., response correctness) and content-specific evaluation (e.g., Does the response state that Mount Everest's height is 8,849m?). Therefore, we add to the baseline that checklist questions should be designed considering possible answers to the input.

Checklist Length While previous studies (Lin et al., 2025; Cook et al., 2024) use a fixed number of items in their checklists, we hypothesize that the optimal number of items depends on the task and should be adjusted accordingly. Therefore, we evaluate how performance changes when we generate checklists containing 0.5 and 1.5 times the number of items for given Baseline checklists.

Self-refine Cook et al. (2024) use LLMs to generate both responses and checklists for tasks, then evaluate responses using these checklists and

perform multiple rounds of self-refine on the responses. However, they do not apply self-refine to the checklist generation process itself. In this study, we extend their approach by implementing self-refine for the checklists to improve their quality. Specifically, our checklist generation model generates a Likert scale evaluation and accompanying feedback based on the baseline prompt and uses this feedback to regenerate improved checklists.

Ticking As a representative of existing methods, we use Cook et al. (2024)'s original prompts. This prompt includes several examples and a limit on the number of checklist items, ranging from two to eight. However, since the original paper does not specify the examples they use, we remove them from our implementation.

3.3 Evaluator Models of Different Sizes

Previous studies have used a limited variety of evaluation models. While the InFoBench (Qin et al., 2024) uses LLMs such as GPT-4 for evaluation, their smallest model is vicuna-13b-v1.5 (Chiang et al., 2023), limiting practical applications. Moreover, their analysis includes only a single smaller model without comparing different sizes of the same model or exploring how checklist usage affects correlation across varying model sizes. To address these limitations, we evaluate the usefulness of checklists across eight models ranging from 7B to 32B parameters, including multiple sizes of the same model family, as detailed in Section 4.3.

4 Experiments

To investigate the usefulness of checklist-based automatic evaluation, this study conducts experiments on two tasks: (1) a pairwise comparison task, in which pairs of LLM's responses are judged for relative quality, and (2) a direct scoring task, in which LLM's responses are rated using a Likert scale.

4.1 Dataset

We use datasets with human-annotated LLM responses, which include reliable evaluation labels and cover diverse real-world tasks with multiple subsets. Such datasets are rare, as it is uncommon to find ones that combine both high-quality human evaluation and broad task diversity. The two datasets we employ sufficiently meet these criteria.

Pairwise Comparison For the pairwise comparison task, we use the LLMBar (Zeng et al., 2024) dataset, which comprises eight English subsets, including three major categories: *Adversarial*, *Natural*, and *Processed*. The *Adversarial* subset includes inputs specifically designed to mislead LLMs when used as evaluators, while the *Natural* subset contains inputs collected and modified from existing human preference datasets. the LLMBar dataset exhibits an inter-annotator agreement exceeding 90%, demonstrating its reliability. The *Processed* subset consists of processed versions of three existing datasets (FairEval (Wang et al., 2024), LLMEval-2 (Zhang et al., 2023), and MT-Bench (Zheng et al., 2023)). These datasets have been refined by Zheng et al. (2023) to improve data fairness. Finally, we obtained manual annotations for 885 response pairs.

Direct Scoring For the direct scoring task, we use the InFoBench (Qin et al., 2024) dataset. the InFoBench dataset uses a Likert scale (1 to 5) as its metric and consists of two English subsets: a simpler section (*Easy* subset) and a more challenging section (*Hard* subset). For each input prompt, responses are collected from five distinct language models (GPT-3.5-turbo (Ouyang et al., 2022), GPT-4, Claude-v1 (Bai et al., 2022), Alpaca-7B (Taori et al., 2023), and Vicuna-13B (Chiang et al., 2023)). The responses are then manually annotated by three expert evaluators, who are natural language processing specialists according to prior research. The correlation coefficient is reported as 0.353 for the *Easy* set and 0.519 for the *Hard* set. This dataset consists of the manual evaluation results for five LLMs' responses across 50 tasks, resulting in a total of 250 annotated samples. In this study, we determine the gold label for each sample by rounding the mean of the three manually annotated labels.

4.2 Checklist Generation

Regardless of the variation, each checklist item is formatted to allow for either a *yes* or *no* response. We employ six different checklist generation policies (detailed in Section 3.2) using gpt-4o-2024-08-06 (OpenAI, 2024) as the generation model.

4.3 Automatic Evaluation

To investigate the usefulness of checklists across different model sizes, we evaluate eight models: gpt-4o-2024-08-06 (GPT-4o), Qwen2.5-32B-Instruct (Qwen2.5-32B-it), Qwen2.5-7B-Instruct (Qwen2.5-7B-it) (Yang et al., 2025), Mistral-Small-24B-Instruct-2501 (Mistral-Small-

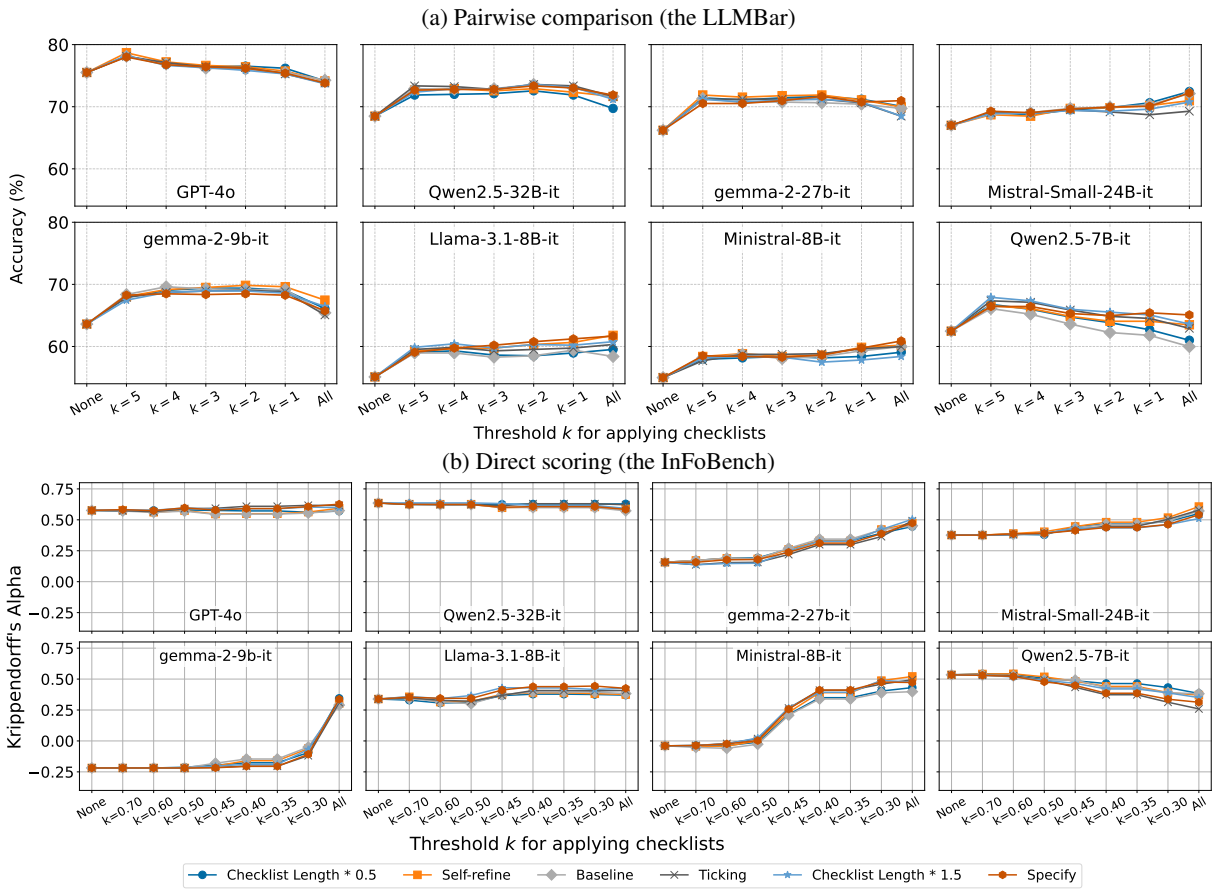

Figure 3: Comparison of accuracy for checklist application method. *None* indicates that the checklist is not used during evaluation, while *All* signifies that the checklist is applied to every evaluation. The parameter k represents the threshold for applying the checklist; the smaller the value of k, the more frequently the checklist is employed.

24B-it), Ministral-8B-Instruct-2410 (Ministral-8B-it) [2], Gemma-2-27b-it, gemma-2-9b-it (Gemma Team, 2024), Llama-3.1-8B-Instruct (Llama-3.1-8B-it) (Dubey et al., 2024). These models represent different parameter sizes and capabilities.

For evaluations without checklists, we use Chain-of-Thought prompting (Wei et al., 2022). We first prompt the model to output the reason for its evaluation, and then obtain the final result. For evaluations with checklists, we first ask the evaluation model to choose the evaluation result for each checklist item from *yes*, *no*, or *n/a*, and then obtain the final evaluation result. *n/a* indicates the item is skipped as it does not apply to the response.

For both tasks, we evaluate each response ten times. To mitigate position bias in pairwise comparison, we use each order five times. The checklist generation prompts are provided in Appendix A.8.

4.4 Evaluation Metrics

For the pairwise comparison dataset, we use accuracy as the evaluation metric for evaluating the performance of our automatic evaluation. The final evaluation result is determined by a majority vote across multiple evaluations; if the votes are evenly split, the outcome is considered a tie. However, because the LLMBar (Zeng et al., 2024) provides only binary labels (win or lose) and does not include a tie label, we assign a score of 0.5 to a tie when calculating accuracy. This allows us to treat accuracy as an expected value under realistic deployment scenarios. This adjustment reflects our goal of evaluating the potential practical benefits of checklist-based evaluation.

For the direct scoring dataset, we use Krippendorff's alpha (Hayes and Krippendorff, 2007) to measure the agreement between automatic and human labels. The final evaluation result of automatic

[2] https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501, https://huggingface.co/mistralai/Ministral-8B-Instruct-2410

evaluation is obtained by taking the mean of ten evaluations and rounding the value.

5 Results

In total, we generate 22,985 checklist items, specifically 21,475 items in the LLMBar and 1,510 from the InFoBench. For detailed statistical analysis of variations and thresholds, see Appendix A.1.

5.1 Identifying When Responses Need Checklist Evaluation

We define a threshold k to determine when to apply checklists based on evaluation inconsistency. Due to the different nature of our evaluation tasks, we use different inconsistency metrics x_{pairwise} and x_{direct} for each setting.

Pairwise Comparison Setting We define the inconsistency value x_{pairwise} as the number of votes the less-preferred response receives. For example, if the evaluations for Response 1 and Response 2 are $[1, 1, 1, 1, 1, 1, 1, 2, 2, 2]$, $x_{\text{pairwise}} = 3$; for $[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]$, $x_{\text{pairwise}} = 5$. We apply checklists only when $x_{\text{pairwise}} \geq k$, where k is selected from $\{1, 2, 3, 4, 5\}$.

Direct Scoring Setting We define the inconsistency value x_{direct} as the standard deviation of these evaluations. For example, if the evaluation labels are $[3, 3, 3, 3, 4]$, $x_{\text{direct}} = 0.4$; for $[2, 3, 3, 3, 4]$, $x_{\text{direct}} = 0.63$. We apply checklists only when $x_{\text{direct}} \geq k$. Based on our observations, most x_{direct} values fall in the range of 0.3 to 0.8, with a notable concentration between 0.3 and 0.5. Therefore, we select k from $\{0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.7\}$.

Figures 3a and 3b show the experimental results. *None* indicates that no checklist is used during evaluation, while *All* denotes that all available checklists are applied. Detailed checklist application rates are provided in Appendix A.2.

Our results demonstrate that the impact of selective checklist application varies across datasets. In the pairwise comparison, we observe that selective checklist application often improves evaluation performance over both the *None* and *All*, for several models, including GPT-4o, Qwen2.5-32B-it, Gemma-2-27B-it, Gemma-2-9B-it, and Qwen2.5-7B-it. In the direct scoring, we observe no improvements from selective checklist usage in direct scoring, where its performance often matches or falls below that of the *None* and *All*.

Bootstrap Sampling We also conduct a bootstrap test to evaluate whether the selective application of checklists leads to improvements. For pairwise comparison, we observe statistically significant differences in 20 out of 48 cases, suggesting that selectively applying checklist items can be beneficial under certain conditions. On the other hand, for direct scoring, we observe no statistically significant differences across any of the six checklist-generation policies evaluated with eight evaluation models, indicating that this approach does not yield measurable improvements under the tested conditions. For detailed settings and results, see Appendix A.3.

5.2 Checklist Generation Policy

Next, we present the results of the checklist generation policy in Figures 3a, 3b, and Appendix A.2. We do not find any specific variation that consistently outperforms others. The best checklist variation depends on the evaluation tasks and evaluator models. For pairwise comparison, Specify works well with GPT-4o and Gemma-2-27b-it, while Ticking suits Ministral-24B-it, Ministral-8B-it, and Llama-3.1-8B-it. In direct scoring, Specify is effective for GPT-4o and Llama-3.1-8B-it, whereas Self-refine performs best with Ministral-24B-it, Ministral-8B-it, and Qwen2.5-7B-it. These findings suggest that checklist methods should be adapted to specific evaluator models and tasks.

Useful and Not Useful Checklist Settings We do not find any checklist generation policies that are consistently superior or inferior across all settings. However, we conduct an in-depth analysis of which policies are *useful* or *useless* for datasets.

Tables 1 and 2 summarize the best and worst performing checklist generation methods, including *None* for each dataset. Both Self-refine and Specify tend to perform well across the two datasets. Specify is useful because it produces checklists containing more detailed information, which helps clarify the evaluation criteria. Self-refine, on the other hand, involves having the LLMs revise the baseline checklist, often resulting in more refined and input-relevant items. This iterative refinement may improve evaluation quality.

Conversely, in the LLMBar, the worst-performing approach is *None*. For 6 out of 8 evaluation models, *None* results in the lowest performance. In the InFoBench, Baseline shows the lowest correlation for half of the evaluation models.

Model	GPT-4o	Qwen2.5-32B-it	gemma-2-27b-it	Ministral-24B-it	gemma-2-9b-it	Ministral-8B-it	Llama-3.1-8B-it	Qwen2.5-7B-it
Best Policy	Specify	Length * (0.5, 1.5)	Specify	Ticking	Self-refine	Ticking	Ticking	Self-refine
Worst Policy	Self-refine	None	None	None	None	None	None	Length * 0.5

Table 1: Best and worst settings of checklist use for each evaluation model in the pairwise comparison.

Model	GPT-4o	Qwen2.5-32B-it	gemma-2-27b-it	Mistral-Small-24B-it	gemma-2-9b-it	Llama-3.1-8B-it	Ministral-8B-it	Qwen2.5-7B-it
Best Policy	Specify	Length * 1.5	Length * 1.5	Self-refine	Length * 0.5	Specify	Self-refine	Self-refine
Worst Policy	Baseline	Baseline	Length * 1.5	Length * 1.5	Specify	Baseline	Baseline	Ticking

Table 2: Best and worst settings of checklist use for each evaluation model in the direct scoring.

These findings suggest that using any checklist benefits pairwise comparison evaluation, whereas the choice of generation method requires more careful consideration for direct scoring evaluation.

5.3 Model Sizes

Finally, we analyze how much correlation with human evaluation improves when small evaluator models use checklists. In the direct scoring dataset, we observe that increasing checklist usage (i.e., lowering the threshold k) contributes to higher correlation with human ratings for some models, such as Gemma-2-27b-it and Mistral-8B-it. In contrast, for other models—including both larger and smaller ones—checklist application does not substantially affect correlation, suggesting a limited contribution to alignment with human evaluation. In the pairwise comparison dataset, checklists only slightly improve the accuracy of small models, indicating limited usefulness since evaluators may already implicitly consider checklist elements.

6 Analysis

To investigate RQ3: *Which checklist items contribute to alignment with human evaluation?*, We conduct ablation and qualitative analyses to identify factors affecting evaluation performance.

6.1 Ablation on Checklist Effectiveness

Experimental Setup We define two types of checklist items. A *positive* item is one whose removal from the checklist leads to a *decrease* in correlation with human evaluation, while a *negative* item is one whose removal leads to an *increase* in correlation. These definitions indicate whether the presence of a checklist item contributes to or hinders alignment with human evaluation. Since ablating each individual checklist item is computationally expensive, we adopt a two-step approach. In the first step, we classify each checklist–not individual items–based on whether its use improves alignment with human evaluation. To measure alignment, we define a score $\Delta\bar{s}_{\text{all}}$ as:

$$\Delta\bar{s}_{\text{all}} = |\bar{s}_{\text{gold}} - \bar{s}_{\text{none}}| - |\bar{s}_{\text{gold}} - \bar{s}_{\text{all}}| \quad (1)$$

where $\bar{s}_{\text{gold}}$ represents the mean score given by human annotators, while $\bar{s}_{\text{all}}$ and $\bar{s}_{\text{none}}$ are mean scores from the model (Qwen2.5-7B-it) with and without checklists, respectively. Based on $\Delta\bar{s}_{\text{all}}$ and predefined thresholds, we classify each checklist as *positive* (it improves alignment with human evaluations) or *negative* (it reduces alignment).

In the second step, we analyze individual checklist items within each group. To quantify the contribution of each item, we define another score:

$$\Delta\bar{s}_{\text{abl}} = |\bar{s}_{\text{gold}} - \bar{s}_{\text{all}}| - |\bar{s}_{\text{gold}} - \bar{s}_{\text{abl}}| \quad (2)$$

where $\bar{s}_{\text{abl}}$ is the mean evaluation score after removing a specific checklist item. If the removal of an item leads to lower alignment, it is classified as a *positive* checklist item; if it leads to higher alignment, it is classified as a *negative* one. For details on the classification and analysis of checklist items, see Appendix A.5.1 and A.5.2.

Quantitative Results We first report the classification result of the checklist items contained in generated checklists for the LLMBar and the InFoBench. In the LLMBar, 53.5% of the checklists used in *positive checklists* are classified as *positive checklist items* (1,079 out of 2,018), while 42.0% of the items in *negative checklists* are classified as *negative checklist items* (3,847 out of 9,157). In the InFoBench, all items in *positive checklists* are classified as *positive checklist items* (56 out of 56), while 40.7% of the items in *negative checklists* are classified as *negative* (599 out of 1,472).

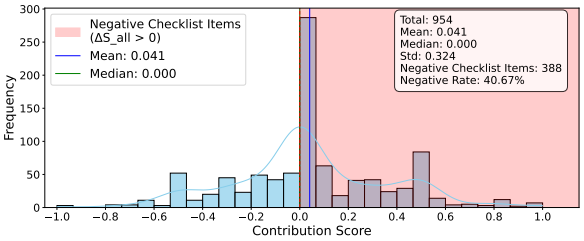

Figure 4: *Negative* checklists ablation results on LLMBar dataset. Each plot shows how removing checklist items impacts correlation with human evaluation. *Negative checklist items* ($\Delta\bar{s}_{\text{abl}}$) are highlighted in red. About 40% of checklist items fall in the *negative* region.

Dataset	Open	Closed
LLMBar	75	10
InFoBench	43	7

Table 3: Open vs. closed classification of questions. the LLMBar questions are sampled as 10% from each of the 8 subsets (85 in total), while all 50 questions from the InFoBench are classified.

We then examine the impact of the *negative checklist items* on alignment with human evaluations, as shown in Figure 4, using results generated by the Baseline. These figures show how removing such items affects evaluation scores compared to using all checklist items. For a comprehensive view of the effects across different checklist types and generation methods, including *positive* and *negative* items, refer to Appendix A.5.3.

These results indicate that a substantial portion of the generated checklist items in *negative checklists* contribute to reduced alignment with human evaluations. However, the impact of such negative items-measured by the change in $\Delta\bar{s}_{\text{abl}}$ is generally small, suggesting that they do not significantly degrade evaluation quality even when present.

6.2 Open vs. Closed Question Classification

We hypothesize that the effectiveness of checklists may depend on the type of question: closed questions may yield more consistent evaluations, whereas open-ended ones can introduce greater variability. To explore this, we manually classify questions in each dataset based on whether their responses tend to converge (*closed*) or diverge (*open*). For the LLMBar, we sample 10% of questions from each of its eight subsets, while for the InFoBench, we analyze all 50 questions. Table 3 shows the classification results. In both datasets, *open* questions outnumber *closed* ones, suggesting that subjective or ambiguous questions are more prevalent. Such questions are more likely to lead to unstable evaluation outcomes and lower agreement with human evaluations, even when using checklists.

# Label	Positive			Negative		
	B	H	G	B	H	G
H	17	3	-	50	46	-
G	29	-	27	75	-	27

Table 4: Checklist quality comparison in the InFoBench. We analyze 274 items (116 human-written, 158 generated). B, H, and G denote items appearing in both, only human, and only generated checklists, respectively. More than half of the items appear in both sets, indicating a notable overlap between human and generated checklist items.

6.3 Overlap Analysis of Human and Generated Checklists in the InFoBench

We manually check the InFoBench checklist items to analyze the extent of overlap between human-written and generated items (274 in total: 116 human-written, 158 generated). Over half of the items appear in both sets, indicating substantial overlap. Here, *both* means checklist items that are semantically equivalent and correspond to the same question. For instance, the generated checklist item *"Does the letter have approximately 250 words?"* closely corresponds to the human-written item *"Is the generated recommendation letter around 250 words? (Output Attribute)"*. Table 4 summarizes the distribution of items by checklist type. Items unique to the generated set—those without overlap with human-written items—often reflect additional perspectives or considerations that, while not explicitly stated in the question, are important for evaluating the response. In contrast, checklist items exclusive to the human-written set tend to focus more on verifying the output format.

6.4 Human Annotation of Checklist Items

To identify checklist item characteristics affecting alignment with human evaluations, we qualitatively analyze 293 items, including 89 *positive* and 102 *negative* items (see Appendix A.6.1). We define six functional labels for *positive* items and four for *negative* ones. Representative examples of generated checklist items are shown in Figure 5. Our annotation shows that 60% of *positive* items explicitly reflect key question elements, aligning with essential response components, while about 30%

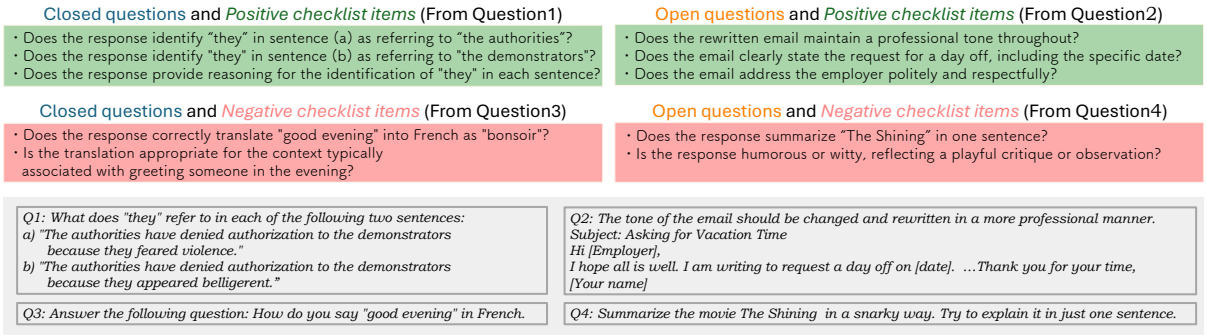

Figure 5: Examples of *positive* and *negative* checklist items by question type (*open* vs. *closed*).

capture important evaluative aspects not explicitly mentioned. For *negative* items, around 10% fail to adequately address response content, suggesting room for improvement; however, over 85% are consistent and deemed usable upon manual review. Furthermore, 77% of these *non-negative* items overlap with criteria created by human-written checklists (see Appendix A.6.2 for details).

6.5 Analysis by Checklist Generation Policy

We also analyze checklist generation policies to examine their characteristics. For example, when handling mathematical problems like "Compute the derivative of $2x^2 + 5x$", the Baseline method generates checklists that break down elements into individual items, such as "Does the response correctly apply the power rule to compute the derivative of $2x^2$?". For this problem, we observe little difference in the generated checklists among the Checklist Length * 0.5, Length * 1.5, and Self-refine methods. In contrast, the Specify method can generate more specific check items that include correct responses while maintaining itemization similar to the Baseline, such as "Did the response simplify the derivative correctly to $4x + 5$?" We also find that different generation methods generate similar checklists. For example, when asking to explain machine learning and its types—supervised, unsupervised, and reinforcement learning—with real-world examples, all methods generate similar items checking basic elements, such as "Does the response elaborate on the differences between supervised, unsupervised, and reinforcement learning?". In contrast, for the task asks how to increase productivity while working from home, the Baseline generates abstract items, such as "Are the suggestions in the response actionable and clear?", while the Checklist Length * 1.5 includes more specific requirements, such as "Is there guidance on setting goals or prioritizing tasks while working from home?". For detailed checklist examples, see Appendix A.7.

Discussion These findings suggest two key directions for future work. First, human evaluations sometimes rely on checklist items with ambiguous criteria or unclear scoring methods that fail to accurately capture response quality. This highlights the need to improve the design of human evaluation protocols. Notably, even *negative* checklist items often overlap with human-written ones, underscoring the difficulty of establishing clear evaluation standards. This observation aligns with Hosking et al. (2024), who highlight inconsistencies and biases in human evaluations. Second, the overlap between generated and human-written checklist items suggests that LLMs can produce reliable and interpretable ones. Combining such generated checklists with human evaluation could improve overall evaluation reliability, rather than relying exclusively on either human or automatic evaluation.

7 Conclusion

We investigate checklist usefulness by focusing on three key questions: determining whether a checklist is necessary for LLM evaluators, designing useful checklists, and analyzing which items are effective. Our experiments show that checklists do not always improve evaluations, and even *negative* items often overlap with human-written ones, revealing limitations in current human evaluations. This highlights the need to reconsider what makes an ideal checklist item that effectively combines human insight and automatic methods, targets relevant criteria, and adapts to different responses. Future work should focus on improving checklist creation and evaluation practices to ensure more reliable and meaningful evaluations.

Limitations

Despite the comprehensiveness of our study, several limitations should be acknowledged. First, while our datasets encompass a diverse range of input tasks, we utilize only a single English dataset for both the pairwise comparison and direct scoring tasks. This constraint may limit the generalizability of our findings across different generative tasks and languages. Second, although we design our checklist generation policies to ensure broad coverage of possible checklist generation methods, there may exist alternative methods that we have not considered, such as those explicitly based on predefined evaluation criteria. Finally, while the models used in our experiments cover multiple families of LLMs, they may still be insufficient to fully capture the necessary features of current LLMs, potentially limiting the scope of our conclusions.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by JST FOREST Grant Number JPMJFR232R. In this work, we used the "mdx: a platform for building data-empowered society". We thank Satoru Katsumata, Hiroaki Sugiyama, Yugo Murawaki, and Sadao Kurohashi for their constructive comments and suggestions that helped improve this paper.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Jonathan Cook, Tim Rocktäschel, Jakob Foerster, Dennis Aumiller, and Alex Wang. 2024. Ticking all the boxes: Generated checklists improve llm evaluation and generation. *Preprint*, arXiv:2410.03608.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. LLM self-correction with DeCRIM: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7773–7812, Miami, Florida, USA. Association for Computational Linguistics.

Gemma Team. 2024. Gemma.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Andrew F. Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for

coding data. *Communication Methods and Measures*, 1(1):77–89.

Tom Hosking, Phil Blunsom, and Max Bartolo. 2024. Human feedback is not gold standard. In *The Twelfth International Conference on Learning Representations*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA. Association for Computational Linguistics.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2025. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797, Albuquerque, New Mexico. Association for Computational Linguistics.

Yukyung Lee, Joonghoon Kim, Jaehee Kim, Hyowon Cho, and Pilsung Kang. 2024. Checkeval: Robust evaluation framework using large language model via checklist. *Preprint*, arXiv:2403.18771.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. 2025. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *Preprint*, arXiv:2411.16594.

Bill Yuchen Lin, Yuntian Deng, Khyathi Raghavi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2025. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In *ICLR*.

Yixin Liu, Alexander Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohan. 2024. Benchmarking generation and evaluation capabilities of large language models for instruction controllable summarization. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4481–4501, Mexico City, Mexico. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

OpenAI. 2024. GPT-4 Technical Report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff 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 Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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. Association for Computational Linguistics.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. 2025. Judgebench: A benchmark for evaluating llm-based judges. *Preprint*, arXiv:2410.12784.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large Language Models are not Fair Evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2024. FLASK: Fine-grained language model evaluation based on

alignment skill sets. In *The Twelfth International Conference on Learning Representations*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following. In *International Conference on Learning Representations (ICLR)*.

Xinghua Zhang, Bowen Yu, Haiyang Yu, Yangyu Lv, Tingwen Liu, Fei Huang, Hongbo Xu, and Yongbin Li. 2023. Wider and Deeper LLM Networks are Fairer LLM Evaluators. *Preprint*, arXiv:2308.01862.

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. *Preprint*, arXiv:2306.05685.

A Appendix

A.1 Checklist Statistics

Tables 5 and 6 show the statistical metrics of checklists for the LLMBar and the InFoBench, respectively. Also, if a checklist cannot be obtained due to an API error or a formatting issue in the response, we regenerate it up to three times. To ensure meaningful comparisons, we filter the datasets to include only questions with consistent checklist counts across all evaluation instances.

Variations	Min	Max	Ave	S.D	Sum
Baseline	1	19	4.63	1.51	3,488
Ticking	2	10	4.92	1.02	3,710
Specify	1	16	5.05	1.54	3,807
Length * 0.5	1	10	2.27	0.74	1,714
Length * 1.5	2	28	6.98	2.38	5,266
Self-refine	1	19	4.62	1.55	3,490

Table 5: Statistical breakdown of generated checklists for each version of the LLMBar. We generate checklists for 754 inputs.

A.2 Checklists Application Rate

Figures 6 and 7 present the checklist application rates for different threshold values in pairwise comparison and direct scoring tasks, illustrating how the threshold k influences the proportion of responses evaluated with checklists.

A.3 Bootstrap Sampling

We conduct a bootstrap test to evaluate whether the selective application of checklists leads to improvements. For the bootstrap procedure, we perform 1,000 resampling iterations, fix the random seed to 42, and determine statistical significance based on 95% confidence intervals.

Variations	Min	Max	Ave	S.D	Sum
Baseline	2	9	4.9	1.64	245
Ticking	3	9	5.3	1.32	265
Specify	2	9	5.32	1.69	266
Length * 0.5	1	4	2.46	0.85	123
Length * 1.5	3	14	7.34	2.53	367
Self-refine	2	9	4.88	1.65	244

Table 6: Statistical breakdown of generated checklists for each version of the InFoBench. We generate checklists for 250 tasks.

For pairwise comparison, we observe statistically significant differences in 20 out of 48 cases. The detailed results for each model are presented below.

- GPT-4o: Shows statistically significant differences for all checklist-generation policies (6/6).
- Qwen2.5-32B-it: Shows significant differences for Baseline, Length$\times$0.5, and Self-refine (3/6).
- Gemma-2-27B-it: Shows no significant difference for any checklist-generation policies (0/6).
- Gemma-2-9B-it: Shows significant differences for all checklist-generation policies (6/6).
- Qwen2.5-7B-it: Shows significant differences for all checklist-generation policies except Ticking (5/6).
- The other three models (Mistral-Small-24B-It, Ministral-8B-It, and Llama-3.1-8B-It) do not show any significant differences (0/6 for each model).

A.4 Checklist Retention Rates after Filtering

The filtering process results in different checklists retention rates across our datasets, as shown in Table 7. In the LLMBar dataset, approximately 90% of checklists are retained for the Ticking and Specify checklist policies, while other categories experience a significant reduction to around 25% of the original checklist count. In contrast, the InFoBench dataset maintains 100% of checklists across all

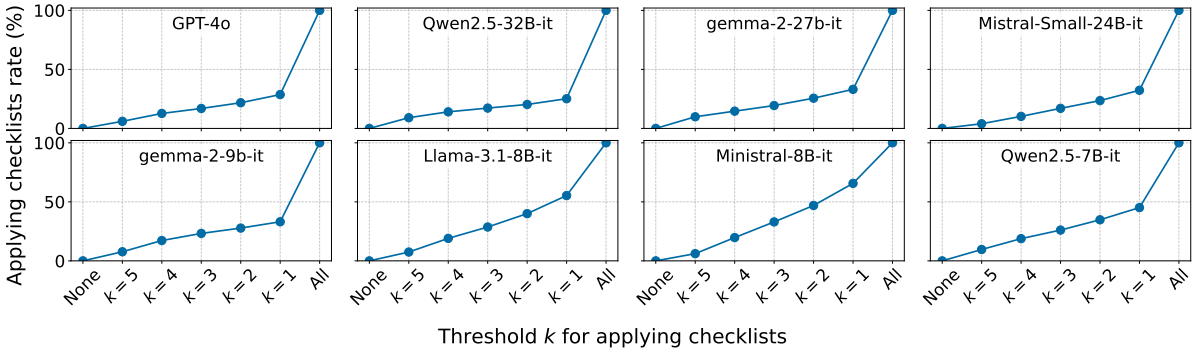

Figure 6: Checklists application rate in the pairwise comparison (LLMBar).

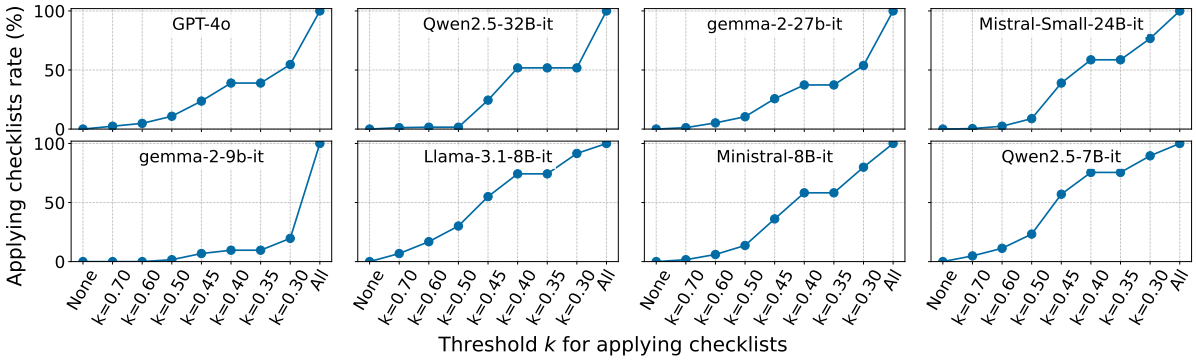

Figure 7: Checklists application rate in the direct scoring (InFoBench).

policies, indicating more consistent checklist application in this dataset. A detailed breakdown of the checklist classification can be found in Figures 8 and 9. In the LLMBar dataset, both *positive* and *negative* checklists each constitute approximately 20% of the total checklists, with the remainder falling into the *neutral* category. The InFoBench dataset shows a different distribution, with *positive* and *negative* checklists each representing only about 2% across most checklist policies. The Length * 1.5 and Self-refine policies stand out as exceptions, with negative checklists surpassing 5% in these cases.

A.5 Ablation Checklist

A.5.1 Selecting Checklist for Ablation

We determine which checklists to use for the ablation of checklist items. To this end, we classify each checklist as:

- **Positive Checklists**: $\Delta\bar{s}_{\text{all}} \geq$ threshold (significantly improves accuracy)
- **Negative Checklists**: $\Delta\bar{s}_{\text{all}} \leq -$threshold (significantly degrades accuracy)

We set different threshold values for each dataset: 0.3 for the pairwise comparison dataset and 1.5 for the direct scoring dataset. We then use the selected positive and negative checklists for checklist item ablation.

A.5.2 Ablation Checklist Items

Based on $\Delta\bar{s}_{\text{abl}}$, we select the final positive and negative checklist items as follows:

- **Positive Checklist Item**: checklist item with $\Delta\bar{s}_{\text{abl}} < 0$, indicating that removing this checklist item degrades performance compared to using all checklists.
- **Negative Checklist Item**: checklist item with $\Delta\bar{s}_{\text{abl}} > 0$, indicating that removing this checklist item improves performance compared to using all checklists.

A.5.3 Results of Checklists After Ablation

Figure 10 shows that positive checklist items predominantly cluster around the 0.0 score, indicating their limited impact on evaluation performance. Similarly, Figure 11 illustrates that nearly half of the negative checklist items have final scores between 0.0 and 0.1, suggesting that their negative

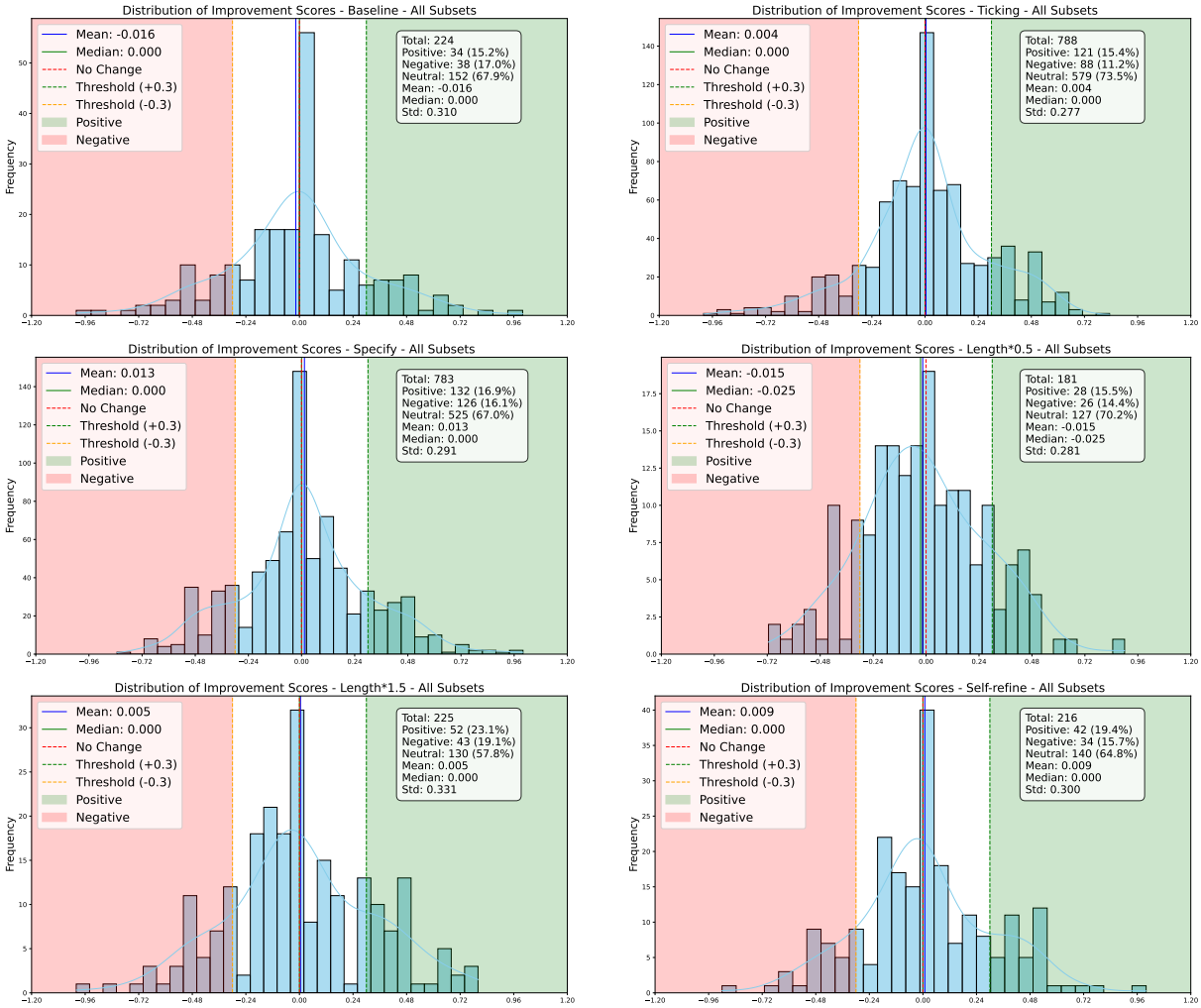

Figure 8: This figure presents the improvement scores computed for LLMBar across different policies. For each policy, the positive and negative checklists respectively comprise approximately 20% of the total items.
The subfigures are arranged as follows: the top row shows the Baseline, Ticking, and Specify checklist policies; the bottom row shows the Length * 0.5, Length * 1.5, and Self-refine policies.

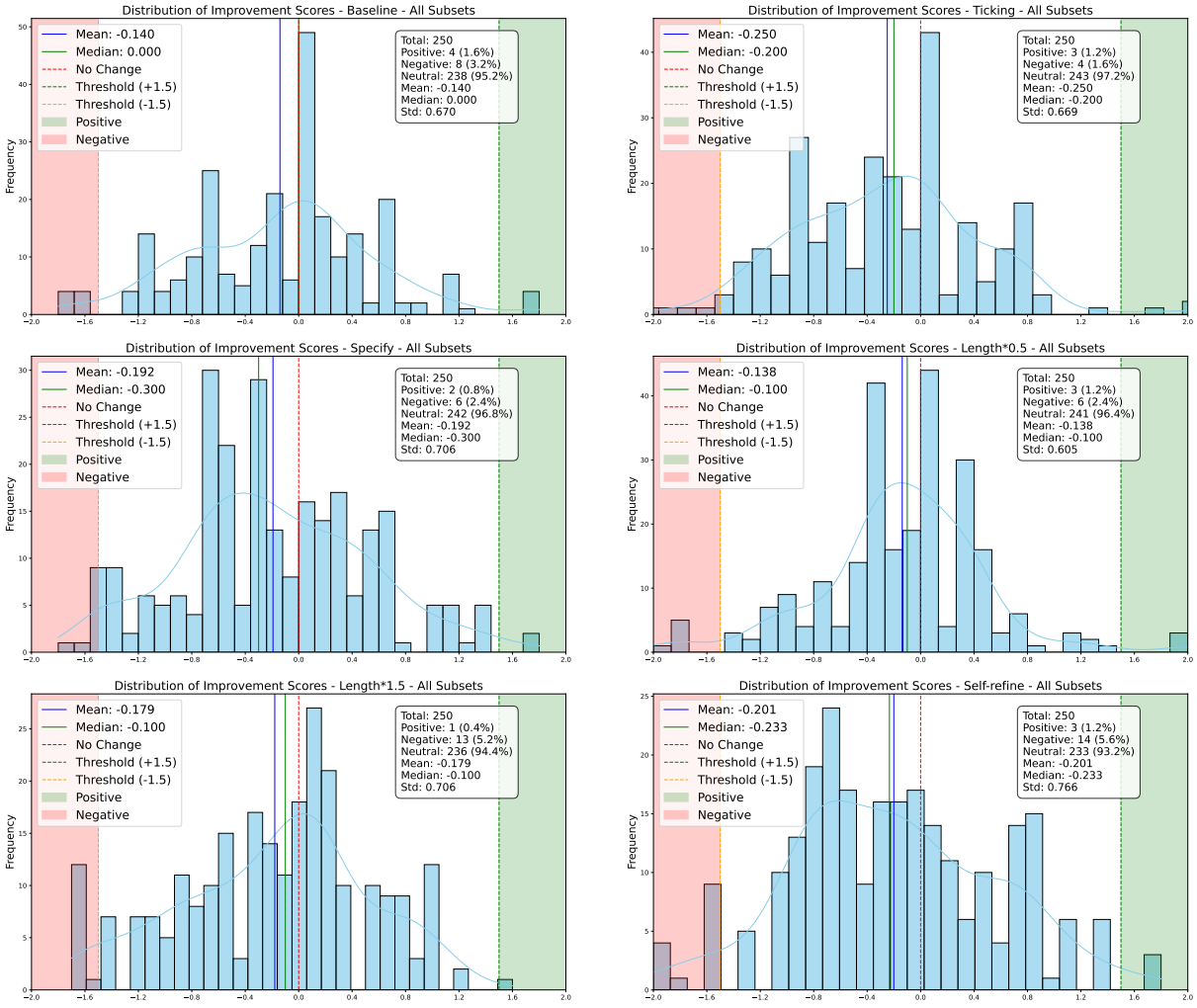

Figure 9: This figure shows improvement scores calculated on InFoBench. For each evaluation policy, approximately 5% of checklists are classified as positive or negative.
The subfigures are arranged as follows: the top row shows the Baseline, Ticking, and Specify checklist policies; the bottom row shows the Length * 0.5, Length * 1.5, and Self-refine policies.

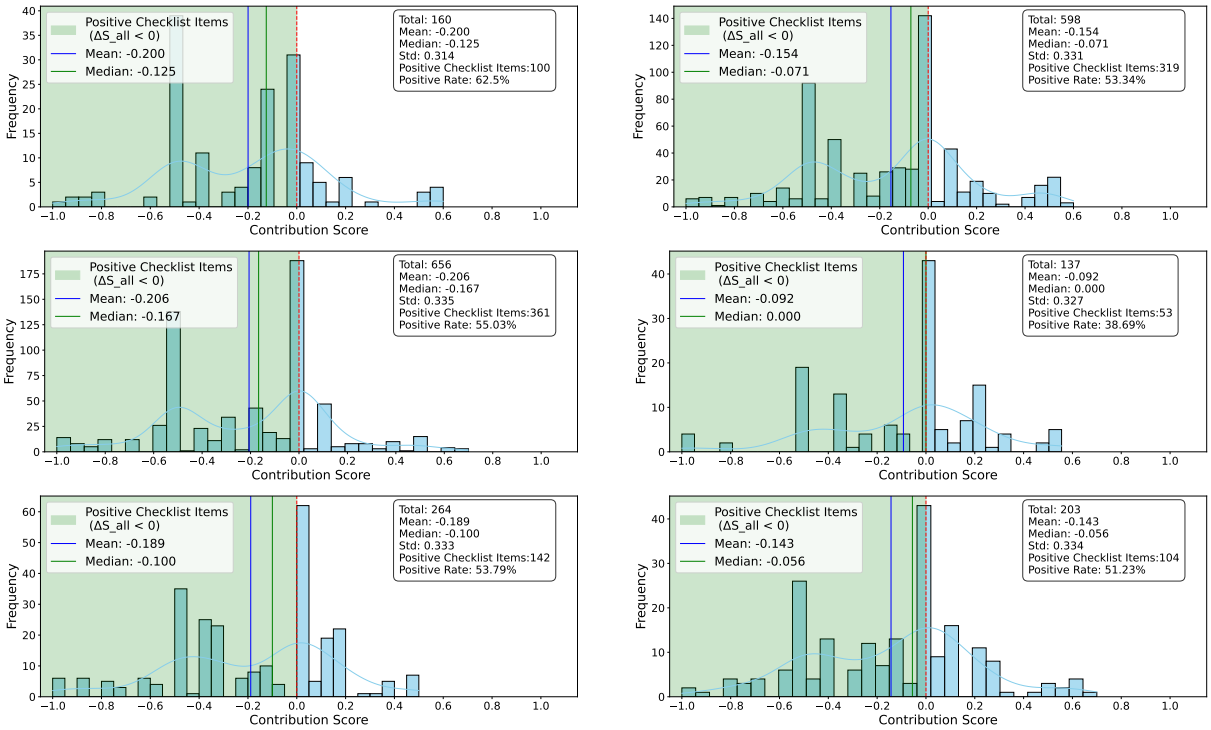

Figure 10: Positive checklist ablation results on the LLMBar dataset. Each plot shows how removing checklist items impacts correlation with human evaluation. *Positive* checklist items ($\Delta\bar{s}_{\text{abl}} < 0$) are highlighted in green. Most scores lie between -0.1 and 0.1. The six generation policies—Baseline, Ticking, Specify, Length*0.5, Length*1.5, Self-Refine—are arranged top-left to bottom-right. All but Length0.5 have over 50% *positive* checklist items; Length0.5 falls below 40%.

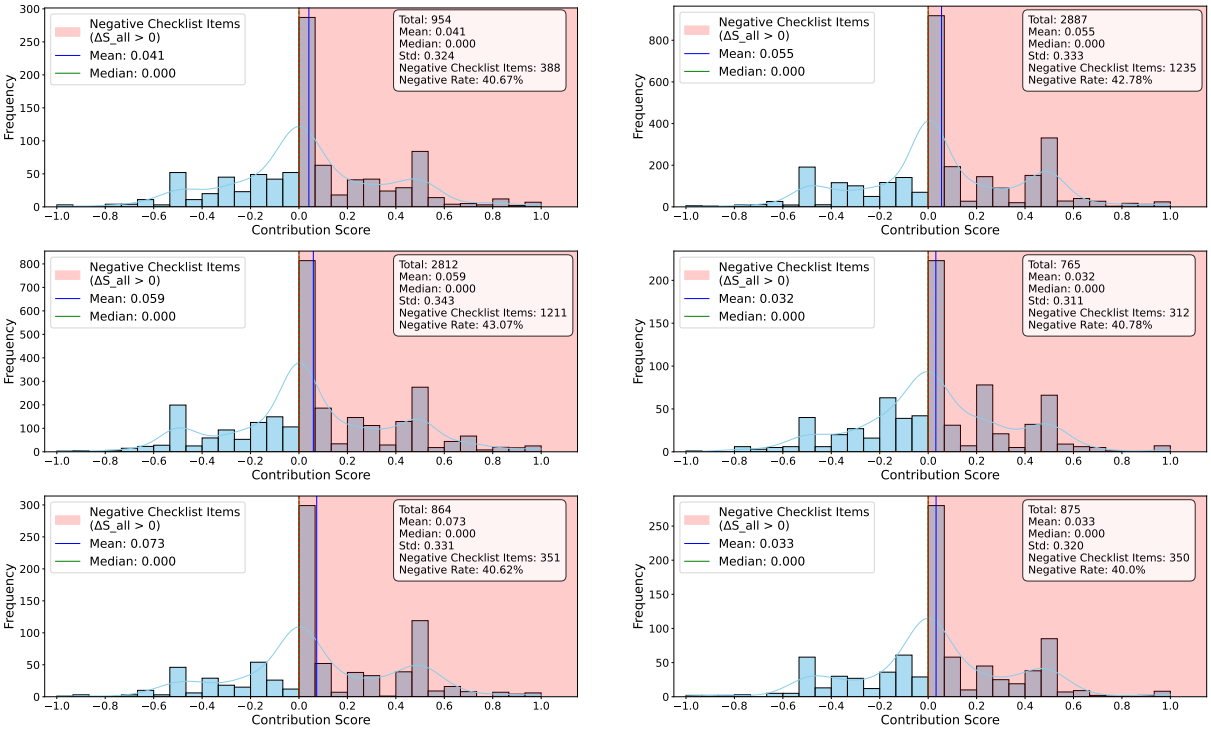

Figure 11: Negative checklist ablation results on the LLMBar dataset. Each plot shows how removing checklist items impacts correlation with human evaluation. *Negative* checklist items ($\Delta\bar{s}_{\text{abl}} > 0$) are highlighted in red. Most scores lie between -0.1 and 0.1. The order of plots matches that of the *positive* checklist items. For all checklist generation policies, the proportion of checklists in the final negative region is around 40%.

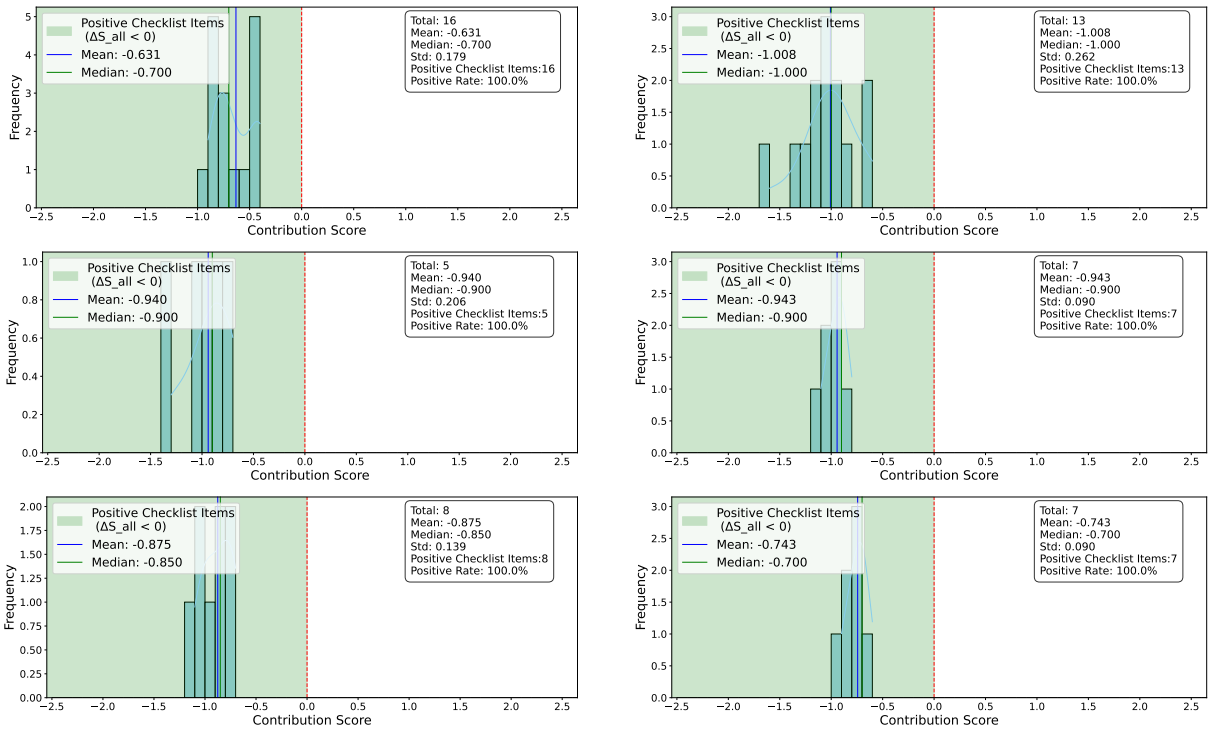

Figure 12: Positive checklist ablation results on the InFoBench dataset. Each plot shows how removing checklist items impacts correlation with human evaluation. *Positive* checklist items ($\Delta\bar{s}_{\text{abl}} < 0$) are highlighted in green. All checklists are *positive* checklist items.

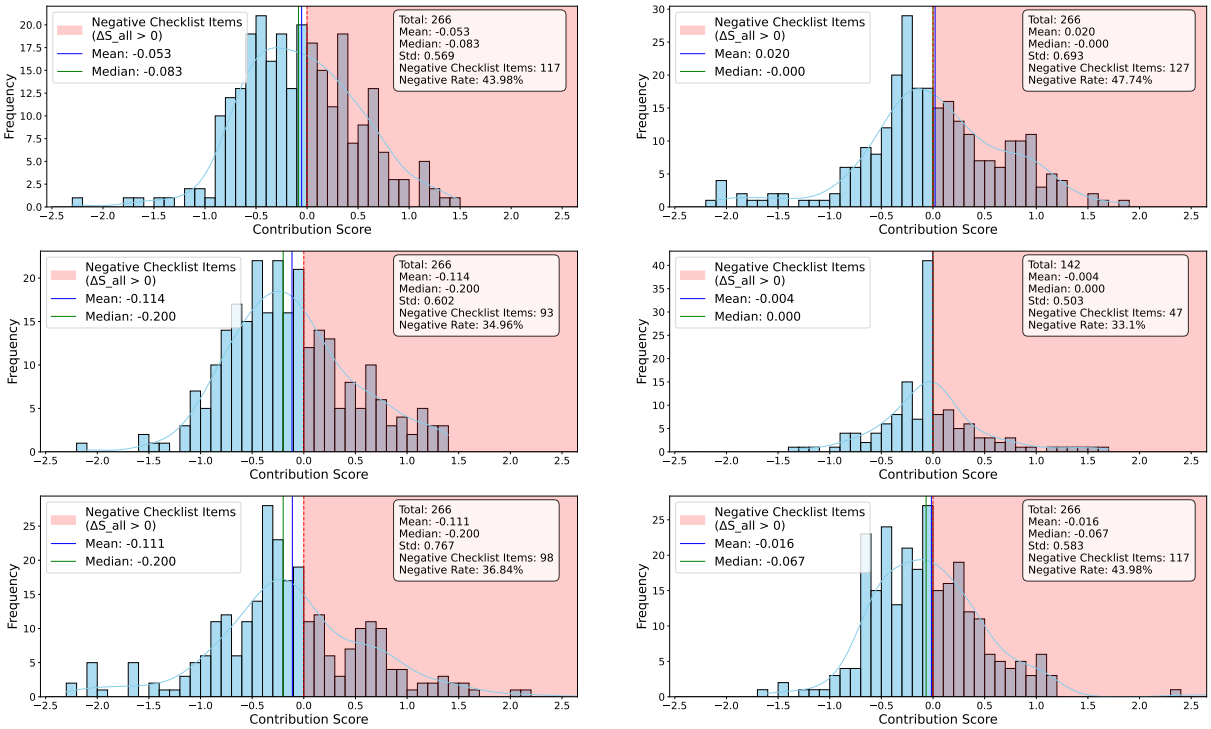

Figure 13: Negative checklist ablation results on the InFoBench dataset. Each plot shows how removing checklist items impacts correlation with human evaluation. *Negative* checklist items ($\Delta\bar{s}_{\text{abl}} > 0$) are highlighted in red. Most scores lie between -0.5 and 0.5. For all checklist generation policies, the proportion of checklists in the final negative region is from 30% to 50%.

Checklist Policy	Original	Verified	Reduced	Verified Ratio (%)	Dataset
Baseline	3488	879	2609	25.2	LLMBar
Ticking	3722	3365	357	90.4	LLMBar
Specify	3821	3467	354	90.7	LLMBar
Length * 0.5	1714	391	1323	22.8	LLMBar
Length * 1.5	5266	1349	3917	25.6	LLMBar
Self-refine	3490	862	2628	24.7	LLMBar
Baseline	245	245	0	100	InFoBench
Ticking	265	265	0	100	InFoBench
Specify	266	266	0	100	InFoBench
Length * 0.5	123	123	0	100	InFoBench
Length * 1.5	367	367	0	100	InFoBench
Self-refine	244	244	0	100	InFoBench

Table 7: Checklist verification results for each method on the LLMBar and the InFoBench, including the number of original, verified, and reduced checklists, and the reduced ratio for each checklist policy. The results show significant variation in checklist reduction, with Ticking and Specify methods achieving the highest reduction ratios on both datasets, while other methods like Length*0.5 show lower reductions.

impact is also minimal. Across different policies, negative checklist items consistently represent approximately 40% of all negative checklists.

However, the proportion of positive checklists that qualify as positive checklist items varies significantly across policies. For example, in the Baseline, over 60% of positive checklists (100 out of 160), whereas in the Length * 0.5, less than 40% meet this criterion (53 out of 137).

In the InFoBench dataset, we identify 56 items as positive checklists and 1472 items as negative checklists in Figure 12 and 13. Notably, all positive checklists qualify as positive checklist items, while 599 items (40.7%) qualify as negative checklist items. The majority of these negative checklist items have final scores between -0.5 and 0.5. Across different policies, negative checklist items represent between 30% and 50% of all negative checklists. Almost all positive checklist items in the InFoBench dataset demonstrate final scores below -0.5, suggesting they consistently bring evaluation results closer to gold labels, albeit by a small margin.

A.6 Qualitative Analysis

A.6.1 Threshold Determination for Qualitative Analysis of Checklist Effectiveness

We detail the specific criteria used to identify checklists with a substantial impact on evaluation performance in each dataset. These thresholds serve as the basis for our qualitative analysis.

For the LLMBar dataset, we analyze the following:

- *Positive* checklist items with $\Delta\bar{s}_{\text{abl}} < 0.9$
- *Negative* checklist items with $\Delta\bar{s}_{\text{abl}} > 0.9$

Applying these stringent criteria, we identify 33 items (1.6% of checklists) and 102 items (2.6% of negative checklists) as having a substantial impact on evaluation.

For the InFoBench dataset, we examine the following:

- All positive checklist items *Negative* checklist items with $\Delta\bar{s}_{\text{abl}} > 0.9$

These thresholds yield 56 items (All positive checklist items) and 102 items (6.9% of negative checklists) that show notable influence on evaluation quality.

A.6.2 In-Depth Qualitative Analysis of Generated Checklist Items

We conduct a qualitative analysis focusing on checklist items that significantly impact evaluation performance, aiming to identify which characteristics contribute to alignment with human evaluation. Based on this analysis, we identify six labels for *positive* checklist items and four labels for *negative* ones.

The following are the *positive* labels:

- **Explicit Focus (Explicit)**: The checklist item clearly states which aspect of the question the response should focus on.
- **Implicit Focus (Implicit)**: The focus is not explicitly stated in the item, but it reflects an important aspect necessary for answering or evaluating the question.

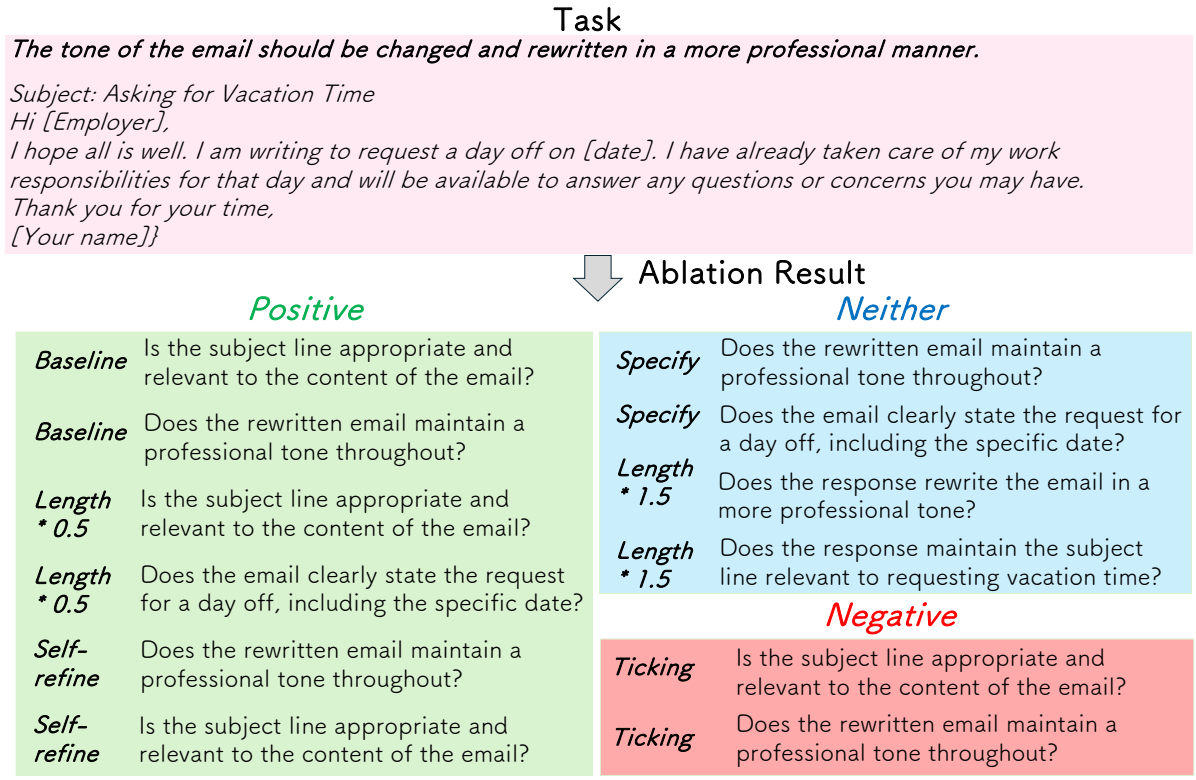

Figure 14: All checklist policies' checklist items. We use Qwen2.5-7B-it as the evaluation model and select the following task from the InFoBench. In the ablation result, we categorize outputs as *Positive*, *Negative*, or Neither, and we show one example. For this task, the Baseline, Length*0.5, and Self-refine variants produce outputs labeled as *Positive*. In contrast, only the Ticking variant produces *Negative* checklist items.

- **Proposal Answer (Ans)**: The item encourages or expects a concrete answer, such as a proposal or opinion, in the response.
- **Clarity**: The item evaluates how easy the response is to understand, such as "Is the response informative and provides a clear explanation?".
- **Additional Content (Add)**: The item includes aspects not strictly required to answer the question, but still useful for checklist-based evaluation.
- **Tone**: The item assesses the appropriateness of the response's tone or style, such as "Is the language clear and formal, appropriate for a legal notice?".

The following are the *negative* labels:

- **Non-negative**: Although classified as a *negative* item, it is still reasonably usable as a checklist item for the given question.
- **Limited Content (Limited)**: The item reflects only a narrow or insufficient aspect of the response, failing to adequately capture its quality.
- **Clarity**: The item evaluates how easy the response is to understand, such as "Is the response logically consistent with the analogy format presented in the question?".
- **Additional Content (Add)**: The item includes aspects not strictly required to answer the question, but still useful for checklist-based evaluation.

Tables 9 and 10 present the distribution of *positive* and *negative* checklist item labels for the LLMBar and the InFoBench, respectively. These tables illustrate how frequently each label type appears in the generated checklists.

A.7 Generate Checklist Example

Table 8 shows an analysis of evaluators and checklists in the InFoBench. This example illustrates how the evaluation results change for multiple evaluators when comparing the cases without a checklist (N) and with a checklist (C), in relation to human labels (H). It also presents the corresponding

Evaluator	Task	H	N	C	Checklist
Qwen2.5-7B-it	Identify the programming language used to write the given code. if (20 > 18) printf("20 is greater than 18");	5	5	3	☐ Does the response correctly identify the use of the 'printf' function as characteristic of the C programming language? ☐ Does the response recognize the syntax of the conditional statement as typical of C-style languages?
gemma-2-27b-it	A confirmation email should be written appropriately for the situation. A meeting has been scheduled, and the sender expects the other to review the slides.	5	3	5	☐ Does the response confirm the details of the scheduled meeting? ☐ Does the response mention that the recipient is expected to review the slides before the meeting?
Ministral-8B-it	Think of alternatives and paraphrases for the underlined word. what we have _expected	5	3	5	☐ Does the response provide alternatives to the word "expected"? ☐ Does the response offer paraphrases that fit in the context of "what we have expected"?

Table 8: Analysis of evaluator and checklists in InFoBench: examples of models with significant changes in correlation when using checklists (H: human labels, N: without checklists, C: checklists applied to a response).

Pos	Explicit	Implicit	Ans	Clarity	Add	Tone
LLMBar	18	5	3	2	5	0
InFoBench	33	20	0	0	2	1
Sum	51	25	3	2	7	1
Rare(%)	57.3	28.1	3.4	2.2	7.9	1.1

Table 9: Label distribution of checklist items classified as *positive*. The majority (≈60%) are **Explicit Focus (Explicit)** items, clearly aligning with elements explicitly stated in the question. About 30% are **Implicit Focus (Implicit)**, reflecting important but implicit evaluation criteria.

Neg	Non-negative	Limited	Clarity	Add
LLMBar	85	11	3	3
InFoBench	92	10	0	0
Sum	177	21	3	3
Rare(%)	86.8	10.3	1.5	1.5

Table 10: Distribution of textitnegative checklist item labels. While more than 85% of the checklist items labeled as **Non-negative** are still aligned with the question and valid upon manual inspection, about 10% are found to be **Limited Content (Limited)** in evaluating the response, suggesting room for improvement in checklist quality.

checklists used in each case. Qwen2.5-7B-it shows lower evaluation performance when using a checklist, reducing its alignment with human judgments. In contrast, Gemma-2-27B-it and Mistral-8B-it improve their alignment with human evaluations when using checklists.

Figure 14 shows all the checklist policies' checklist items. We use Qwen2.5-7B-it as an evaluation model and one of the *open* questions as a task. The column labeled "Ablation Result" reports the outcome of ablation studies conducted on individual checklist items. Items marked as *Positive* contribute to alignment with human evaluations, while those marked as *Negative* do not. Items labeled *Neither* show no clear effect in either direction. For each policy, we provide two representative examples to illustrate the effects.

A.8 Prompts for Checklist Generation and Response Evaluation

Figures 15, 16, 17, 18, and 19 present the prompt used for generating checklists(Baseline, Specify, Length,Self-refine, and Ticking). We use GPT-4o as the generation model.

Figures 20, 21, 22, and 23 present the prompts used for evaluating responses. Figures 20 and 21 show the evaluation prompts for the pairwise comparison datasets, while Figures 22 and 23 show the prompts for the direct scoring datasets.

```
In order to evaluate the AI's response to the input, please create a checklist based on the input.

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 input 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 input where appropriate.

You should always analyze the input before providing an evaluation checklist.
Please enclose each checklist item in double square brackets and output in the following format:

Analysis: Please state the analysis for the input
Checklist:
  - [[Item 1]]
  - [[Item 2]]

[Start of Input]
{question}
[End of Input]
```

Figure 15: Prompt used in the Baseline.

```
In order to evaluate the AI's response to the input, please create a checklist based on the input.

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 input 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 input where appropriate.
  - **Be considerate of expected answers**, meaning that checklist questions should be designed while taking into account
      possible answers to the input.

You should always analyze the input before providing an evaluation checklist.
Please enclose each checklist item in double square brackets and output in the following format:

Analysis: Please state the analysis for the input
Checklist:
  - [[Item 1]]
  - [[Item 2]]

[Start of Input]
{question}
[End of Input]
```

Figure 16: Prompt used in the Specify.

```
In order to evaluate the AI's response to the input, please create a checklist based on the input.

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 input 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 input where appropriate.
  - **Be comprised of {n} items**, meaning that the checklist must contain {n} questions.

You should always analyze the input before providing an evaluation checklist.
Please enclose each checklist item in double square brackets and output in the following format:

Analysis: Please state the analysis for the input
Checklist:
  - [[Item 1]]
  - [[Item 2]]

[Start of Input]
{question}
[End of Input]
```

Figure 17: Prompt used in the Checklist Length. First, the number of checklist items for the Baseline method is calculated. Then, the prompt instructs to generate a checklist with the number of items multiplied by 0.5 or 1.5.

```
Please refine the given checklist for evaluating the AI's response according to the following steps:

1. Evaluate each checklist item based on the requirements below.

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 input 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 input where appropriate.

2. Rate each checklist item on a 5-point size (1 = lowest, 5 = highest) and provide feedback. Present your evaluation using the format below:

Checklist Evaluation:
  1. [Item 1] - [Rating]: [Feedback]
  2. [Item 2] - [Rating]: [Feedback]

3. Refine the checklist based on this feedback. Enclose each checklist item in double square brackets and output them in the following format:

Refined Checklist:
  - [[Item 1]]
  - [[Item 2]]

[Start of Input]
{question}
[End of Input]

[Start of Base Checklist]
{checklist}
[End of Base Checklist]
```

Figure 18: Prompt used in the Self-refine. First, LLM generates checklists using the Baseline method. Next, LLM outputs a Likert scale (1-to-5) evaluation and its rationale for the generated checklists. Finally, based on this rationale, LLM regenerates checklists.

```
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 analyze the INSTRUCTION before providing an evaluation checklist.
## Response Format
Analysis: xxx
Checklists:
 - [[item1]]
 - [[item2]]

## Real Task

### INSTRUCTION
{question}

### Response
Please analyze the instruction and provide 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.
```

Figure 19: Prompt uses in the Ticking. We use TICKing’s (Cook et al., 2024) prompt. Since their paper does not contain examples that they use, we remove the specific in-context examples to use this prompt.

```
Please evaluate the answers according to the following procedure:

1. Evaluate the two responses to the input.
2. First, state the reason for your evaluation. Then decide which answer is better: 1 or 2.
3. Output your response in the following JSON format:
“““ json
{{
   “reason“: “Please state the reason for evaluation“,
   “judge“: 1,
}}
“““
4. Do NOT output anything other than the JSON format.

[Start of Input]
{question}
[End of Input]

[Start of Response 1]
{response_1}
[End of Response 1]

[Start of Response 2]
{response_2}
[End of Response 2]
```

Figure 20: Prompt used for pairwise response comparison, conducted **without** any checklist-based evaluation criteria.

```
Please evaluate the answers according to the following procedure:

1. Evaluate the two responses to the input.
2. Use the given checklist for your evaluation.
3. For each item in the checklist, output “yes”, “no”, or “n/a” if it does not apply.
4. Compare the two responses based on the checklist.
5. Decide which answer is better: 1 or 2.
6. Output your response in the following JSON format:
“““json
{{
   {{
   “checklist”:
      “response_1”: {{
      “1”: “yes”,
      “2”: “no”,
      “3”: “n/a”
      }},
      “response_2”: {{
      “1”: “no”,
      “2”: “n/a”,
      “3”: “yes”
      }}
   }}
   “judge”: 1,
}}
”””
7. Do not output anything other than the JSON format.

[Start of Input]
{question}
[End of Input]

[Start of Checklist]
{checklist}
[End of Checklist]

[Start of Response 1]
{response_1}
[End of Response 1]

[Start of Response 2]
{response_2}
[End of Response 2]
```

Figure 21: Prompt used to evaluate response pairs using checklist-based criteria.

Please evaluate the following response according to the procedure below:

1. Evaluate the response to the input on a size of 1 (lowest) to 5 (highest).
2. First, state the reason for your evaluation. Then provide the overall rating (1-5).
3. Use the following guidelines for each rating:
   Rating-1 (Very Bad): The model's response is entirely irrelevant to the input posed. It demonstrates no understanding of the requested task.
   Rating-2 (Bad): While there is a minimal degree of relevance to the original input, the response is predominantly incorrect, confusing, or both.
   Rating-3 (Okay): The response shows some relation to the posed input but lacks critical details or includes inaccuracies.
   Rating-4 (Good): This rating is given when the response is mostly accurate and comprehensive. However, it might include slight errors or omit minor details.
   Rating-5 (Excellent): A response receiving this rating is entirely accurate, detailed, and aligns perfectly with the requirements of the input.
4. Output your response in the following JSON format:
```json
{{
   `reason': `Explain the rationale for the rating',
   `rating': 3
}}
```
5. Do NOT output anything other than the JSON format.

[Start of Input]
{question}
[End of Input]

[Start of Response]
{response}
[End of Response]

Figure 22: Prompt used for direct scoring evaluation without checklists.

Please evaluate the following response according to the procedure below:

1. Evaluate the response to the input on a size of 1 (lowest) to 5 (highest).
2. Use the given checklist for your evaluation.
3. For each item in the checklist, output “yes,” “no,” or “n/a” if it does not apply.
4. Evaluate the response based on the checklist.
5. Use the following guidelines for each rating:
   Rating-1 (Very Bad): The model’s response is entirely irrelevant to the input posed. It demonstrates no understanding of the requested task.
   Rating-2 (Bad): While there is a minimal degree of relevance to the original input, the response is predominantly incorrect, confusing, or both.
   Rating-3 (Okay): The response shows some relation to the posed input but lacks critical details or includes inaccuracies.
   Rating-4 (Good): This rating is given when the response is mostly accurate and comprehensive. However, it might include slight errors or omit minor details.
   Rating-5 (Excellent): A response receiving this rating is entirely accurate, detailed, and aligns perfectly with the requirements of the input.
6. Output your response in the following JSON format:
“““json
{{
   “checklist”: {{
      “1”: “yes”,
      “2”: “no”,
      “3”: “n/a”
   }}
   “rating”: 3,
}}
“““
7. Do NOT output anything other than the JSON format.

[Start of Input]
{question}
[End of Input]

[Start of Checklist]
{checklist}
[End of Checklist]

[Start of Response]
{response}
[End of Response]

Figure 23: Prompt used for direct scoring evaluation with checklists.