Auditing the Judge: Human-Grounded Bias Discovery, Quantification, and Mitigation in LLM Judges

Anonymous Authors[1]

Abstract

While Large Language Models (LLMs) are increasingly deployed as automated evaluators in evaluation and training pipelines, their judgements are often affected by systematic biases that conflict with human preferences. While prior work has identified several known biases and proposed methods for their detection and mitigation, they lack strong grounding in human evaluation preferences, which is essential to ensuring that the identified biases correspond to actual human judgment behavior. Moreover, they rely heavily on pre-discovered bias lists, overlook bias strength, and depend on costly interventions. In this work, we propose HUB-J, an integrated framework grounded in human evaluation preferences that detects, quantifies, and mitigates biases in LLM-as-a-Judge systems. Our approach leverages human–LLM judgement disagreement cases to automatically discover interpretable bias factors, and utilizes agreement cases to quantify bias strength through controlled input modifications and resulting shifts in model decisions. Finally, building on these quantified biases, we introduce a lightweight, training-free regression-based mitigation strategy that corrects bias-influenced judgments by removing the estimated bias effects. Empirical results show that HUB-J uncovers both known and novel bias factors, reveals meaningful differences in model susceptibility, and consistently reduces bias-driven decision flips while generalizing across models.

1. Introduction

With recent advancements in Large Language Models (LLMs), LLM-based evaluation (often referred to as *LLM-as-a-Judge*) has been increasingly adopted in both evaluation and training pipelines, making their reliability an important problem (Zheng et al., 2023; Ye et al., 2025b; Liu et al., 2023; Ong et al., 2025; Seo et al., 2026). One major concern is systematic evaluation biases (Wang et al., 2024a; Zeng et al., 2024; Dubois et al., 2024) in LLM judges, where LLM judges prefer one response over another based on factors that do not reflect genuine response quality or human evaluators' preferences. For example, (Zheng et al., 2023) identified position and verbosity biases, where LLM judges favor responses based on their presentation order or length rather than the actual content of the answer. However, manually discovering such biases remains challenging, as they are often implicit and costly to annotate at scale. This has motivated recent studies on automatic bias discovery and analysis with LLM (Lai et al., 2026; Ye et al., 2025a).

However, prior approaches face fundamental limitations. First, in *bias identification*, the exploration of new candidates relies heavily on pre-discovered bias lists or LLM's reasoning, which inherently constrains the search space; moreover, this identification is conducted in an instance-wise and model-specific manner, which limits the generalizability of the discovered biases (Lai et al., 2026). Second, in *bias quantification*, prior studies largely treat bias as a binary phenomenon whether a bias flips the judge's decision, offering limited insight into how strongly each bias manifests in a given judge (Ye et al., 2025a). Third, in *bias mitigation*, many existing approaches (Sant et al., 2024; Yang et al., 2026) depend on model-specific interventions, such as additional training or prompt engineering, which incur substantial training or engineering costs. Overall, these limitations call for an integrated framework that discovers, quantifies, and mitigates biases in a manner explicitly grounded in human evaluation preferences.

Contribution. In this work, we propose HUB-J, a unified framework to 1) automatically discover **Hu**man-grounded **B**ias factors in LLM **J**udges, 2) quantify their model-specific effects, and 3) perform lightweight debiasing. Our key idea is to ground LLM judgments in human preferences by selectively using publicly available human preference datasets: disagreement cases between human labels and LLM judgments are used to identify biases that drive human–

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

LLM judgment gaps, while agreement cases are used to quantify how strongly each bias can affect judgments. Specifically, HUB-J first identifies bias factors from cases where human and model judgments disagree by extracting instance-level features and grouping them into interpretable factors. This allows us to discover biases that emerge from actual disagreement without relying on a fixed list of known biases. Second, HUB-J quantifies a specific bias factor using cases where human and model judgments agree; we define a 1–5 Likert scale for bias intensity and modify the rejected response (*i.e.*, the one neither the human nor the model selected) to reflect different levels of the factor. We then measure whether the judge's decision flips in favor of the modified response, and use this flip rate to estimate each model's susceptibility to the factor and derive a model-specific bias profile. Finally, we propose a lightweight regression-based approach to estimate bias effects and recover counterfactual judgments in their absence.

The empirical results validate the effectiveness of HUB-J across all three stages of the framework. In bias identification, HUB-J successfully discovers bias factors that are strongly supported by disagreement signals, while also revealing novel bias factors underlying judge misalignment. For bias quantification, HUB-J reveals model-specific bias preference profiles, showing that different LLM judges are susceptible to different factors and effective intensity levels. For instance, *Technical Jargon*–a newly discovered, human-interpretable factor not covered by the baseline bias sets–shows that different judges respond to it very differently. Also, an extended experiment with 20 state-of-the-art LLMs further shows that these bias patterns persist across a broader model pool, suggesting that the discovered factors capture generalizable judge-specific vulnerabilities. Finally, our proposed bias mitigation method reduces bias-induced corruption in judgments, correcting deviations of up to 19.8%. In fact, it remains robust across five judges spanning diverse model families and scales. Overall, these results show that HUB-J provides a unified and effective framework for discovering, quantifying, and mitigating LLM-as-a-Judge biases while grounding them in human preferences.

2. Related Work

LLM-as-a-Judge. LLM-as-a-Judge has emerged as a scalable alternative to human evaluation in open-ended generation, especially in cases where reference-based metrics fail to reflect human preferences (Koo & Kim, 2026). Results from MT-Bench and Chatbot Arena show that strong LLMs can operate effectively as pairwise evaluators, motivating their use in model comparison and benchmark development (Zheng et al., 2023). More recent work has developed evaluator models, automatic evaluation protocols, and meta-evaluation benchmarks, including G-Eval, PandaLM, Prometheus, Length-Controlled AlpacaEval, LLMBar, and JudgeBench (Liu et al., 2023; Wang et al., 2024b; Kim et al., 2024; Dubois et al., 2024; Zeng et al., 2024; Tan et al., 2025). Beyond evaluation, preference judgements are also important to alignment pipelines such as reward modeling and reinforcement learning from human or AI feedback (Ouyang et al., 2022; Bai et al., 2022; Lee et al., 2024). Since these signals influence both model selection and optimization, their reliability is becoming increasingly important in practice.

Bias in LLM-as-a-Judge. Prior work shows that LLM judges can be influenced by superficial factors such as response order, length, and model identity rather than underlying response quality, leading to position, verbosity, self-enhancement, and related biases (Raina et al., 2024; Zheng et al., 2023; Wang et al., 2024a; Chen et al., 2024; Panickssery et al., 2024; Jeong et al., 2025). Subsequent studies quantify pre-discovered bias dimensions, evaluate judge robustness with challenging benchmarks, and explore automatic bias discovery or debiased judge training (Ye et al., 2025a; Tan et al., 2025; Zeng et al., 2024; Lai et al., 2026; Yang et al., 2026). Related work also shows that judge preferences can be reverse-engineered or exploited by optimizing against LLM-judge feedback (Alazraki et al., 2025). However, existing approaches often rely on pre-discovered factors, validate biases through local perturbations, or treat bias as a binary flip event (Lai et al., 2026; Ye et al., 2025a). On the other hand, HUB-J discovers human-grounded bias factors from human–LLM disagreement, measures model-specific bias strength via controlled intensity scaling, and reduces their effects without retraining the judge.

3. Method

In this section, we elaborate the full method of HUB-J. The algorithmic details across all stages are in Appendix B.

3.1. Preliminaries

We consider a pairwise human preference dataset

$$\mathcal{D}^{\text{full}} = \{d_i = (x_i, r_i^A, r_i^B, y_i^H)\}_{i=1}^N,$$

where x_i is the prompt, r_i^A and r_i^B are the two candidate responses, and $y_i^H \in \{A, B\}$ is the human preference label. For each instance, we denote the human-preferred and human-rejected responses as

$$r_i^+ = r_i^{y_i^H}, \qquad r_i^- = r_i^{\bar{y}_i^H},$$

where $\bar{y}_i^H$ is the unselected option. For a judge model m with prediction $y_i^m = m(x_i, r_i^A, r_i^B)$, we partition $\mathcal{D}^{\text{full}}$ into per-model agreement and disagreement datasets:

$$\mathcal{D}_m^{\text{agr}} = \{d_i \in \mathcal{D}^{\text{full}} \mid y_i^m = y_i^H\},$$
$$\mathcal{D}_m^{\text{dis}} = \{d_i \in \mathcal{D}^{\text{full}} \mid y_i^m \neq y_i^H\}.$$

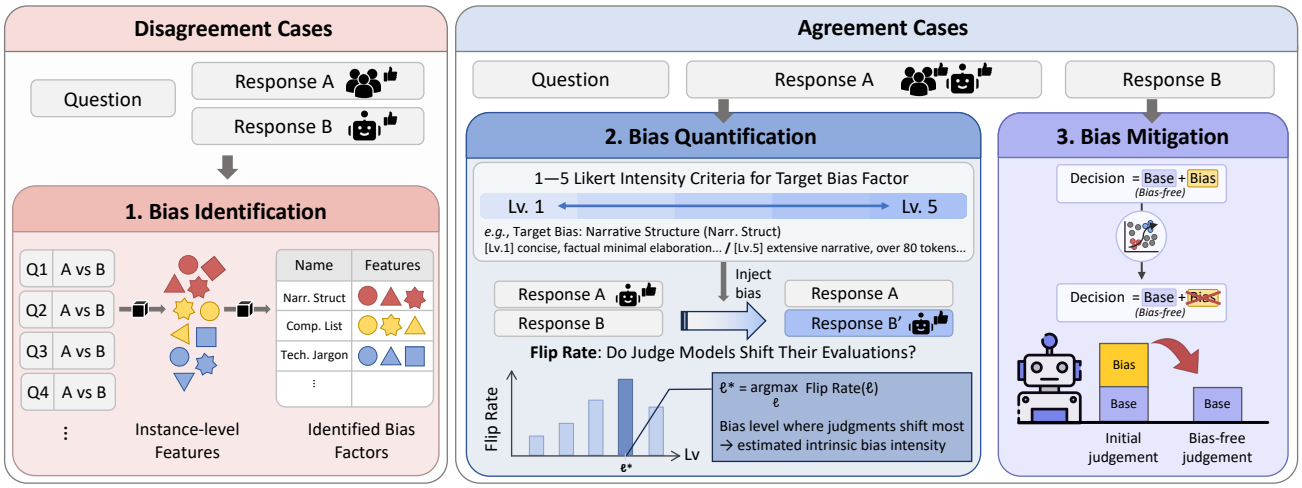


Figure 1. **Overview of HUB-J**, a unified framework for discovering, quantifying, and mitigating human-grounded bias in LLM-as-a-judge systems. (1) *Bias identification* uses human–LLM disagreement cases to extract and cluster instance-level features into bias factors. (2) *Bias quantification* uses agreement cases by injecting each factor into rejected responses across 1–5 Likert levels, defining effective intensity as the point of strongest model decision shift. (3) *Bias mitigation* decomposes judgments into bias-free and bias-attributable parts, enabling bias estimation and correction.

so that, in $\mathcal{D}_m^{\text{dis}}$, the judge m selects the human-rejected response r_i^- over the human-preferred response r_i^+ while, in $\mathcal{D}_m^{\text{agr}}$, both the human and judge m select r_i^+.

This separation naturally suits the two stages of our framework. We use $\mathcal{D}_m^{\text{dis}}$ for *bias identification*, since cases where judge m favors the human-rejected response can expose latent biases of the judge model that drive its decisions away from human preferences. We use $\mathcal{D}_m^{\text{agr}}$ for *bias quantification* because these cases provide a controlled starting point: both humans and judge m originally prefer r_i^+ over r_i^-. We then inject a candidate bias factor into the originally rejected response r_i^- and measure whether judge m switches its preference toward the modified response.

Throughout the following sections, we denote LLM-based generation modules by $G_\bullet$, where the subscript specifies the task performed by each module (*e.g.*, G_{extract} for extracting interpretable features during bias identification stage). The prompt for each module is provided in Appendix H.

3.2. Bias Identification

To identify bias factors that generalize across models, we construct a consensus disagreement subset rather than relying on a single judge m, since using only one judge may introduce model-specific tendencies that limit generalization. Specifically, given a set of state-of-the-art LLM judges $\mathcal{M} = \{m_1, \ldots, m_K\}$, we define a consensus disagreement subset as $\mathcal{D}_{\mathcal{M}}^{\text{dis}} = \bigcap_{k=1}^{K} \mathcal{D}_{m_k}^{\text{dis}}$. This focuses our analysis on cases where multiple LLM judges consistently disagree with human preferences, rather than being driven by model-specific effects. Inspired by (Kim et al., 2026; Wang et al., 2023), we subsequently conduct the following pipeline to identify and construct bias factors.

Feature extraction. For each instance $d_i \in \mathcal{D}_{\mathcal{M}}^{\text{dis}}$, we use a feature extraction module G_{extract} to extract a set of features $\mathcal{F}_i = \{f_{i,j}\}$, where each $f_{i,j}$ captures a possible reason for the discrepancy between human and LLM judge preferences for the given response pair. Then, we aggregate these instance-level features into a global feature pool:

$$\mathcal{F} = \bigcup_{d_i \in \mathcal{D}_{\mathcal{M}}^{\text{dis}}} \mathcal{F}_i.$$

Factorization. Using the extracted feature set $\mathcal{F}$, our goal is to construct a compact set of top-K bias factors that best explain $\mathcal{F}$. To this end, we conduct following processes inspired by RPM (Kim et al., 2026):

Stage 1: Candidate factor initialization. We first apply K-means clustering on the feature set $\mathcal{F}$ to group semantically similar features, where each cluster represents a potential latent bias region. From each cluster, we select the feature closest to the centroid as its representative. All representatives are passed to $G_{\text{construct}}$, which generates a set of candidate bias factors $\mathcal{Z}^{\text{cand}} = \{z_1, \ldots, z_M\}$, together with textual descriptions t_z for each factor. Then, to remove semantically redundant factors in $\mathcal{Z}^{\text{cand}}$, we apply embedding-based similarity filtering. Each factor z is represented by an embedding e_z, and we compute pairwise cosine similarity. We obtain a refined factor set $\tilde{\mathcal{Z}} \subseteq \mathcal{Z}^{\text{cand}}$ such that redundant factors with similarity above threshold $\tau \in [0, 1]$ are removed.

Stage 2: Assignment and coverage estimation. Subsequently, we use an assignment module G_{assign} to map each feature to its most relevant factor or abstains if no factor sufficiently explains it:

$$a = G_{\text{assign}}(\mathcal{F}, \tilde{\mathcal{Z}}), \quad a : \mathcal{F} \to \tilde{\mathcal{Z}} \cup \{\perp\}.$$

For each factor $z \in \tilde{\mathcal{Z}}$, we define its coverage set $\mathcal{C}(z) = \{f \in \mathcal{F} \mid a(f) = z\}$ and the uncovered feature set $\mathcal{F}_{\text{unc}} = \{f \in \mathcal{F} \mid a(f) = \perp\}$.

Stage 3: Greedy top-K selection. Then, we construct the final factor set $\mathcal{Z} = \{z_1, \ldots, z_K\}$ by greedily selecting K factors from $\tilde{\mathcal{Z}}$. At each step, we select:

$$z^* = \arg\max_{z \in \tilde{\mathcal{Z}} \setminus \mathcal{Z}} \Big[\alpha \cdot |\mathcal{C}(z) \setminus \mathcal{U}(\mathcal{Z})| - \lambda_u \cdot |\mathcal{F}_{\text{unc}}|\Big],$$

where $\mathcal{U}(\mathcal{Z}) = \bigcup_{z \in \mathcal{Z}} \mathcal{C}(z)$ denotes the set of features already covered by the current factor set. Hyperparameters α and λ_u control the strength of each term in the objective.

The features not covered by the final selected factors are added to $\mathcal{F}_{\text{unc}}$; then, we repeat Stages 1–3 using the current uncovered feature set $\mathcal{F}_{\text{unc}}$, while retaining and updating the candidate factor pool across iterations. This process continues until either the coverage ratio exceeds threshold ρ or a maximum of T iterations is reached.

3.3. Bias Quantification

Given a bias factor z and its textual description t_z, we quantify how strongly z affects a given judge model. To this end, we inject factor z into the originally rejected response in human-LLM judgement agreement cases and measure whether the judge flips toward the modified response.

Bias injection with intensity criteria. To control the strength of the bias factor, we first convert the factor description t_z into a five-level intensity criterion where lower intensity indicates weaker bias:

$$\mathcal{C}_z = G_{\text{crit}}(z, t_z) = \{c_{z,\ell}\},$$

where $c_{z,\ell}$ specifies how the bias factor z should be expressed at intensity level $\ell \in L$, $L = \{1, 2, 3, 4, 5\}$. Each criterion pairs a description with factor-specific quantifiable anchors (*e.g.*, token ranges). By grounding each level in observable response properties, these anchors allow us to inject the same factor at controlled strengths, ordered from the weakest to the strongest manifestation.

Then, for each agreement instance $d_i \in \mathcal{D}_m^{agr}$ where both human and judge m select r_i^+, we rewrite the rejected response r_i^- based on target factor z and intensity level ℓ:

$$\tilde{r}_{i,z,\ell}^- = G_{\text{inj}}(x_i, r_i^-, z, t_z, c_{z,\ell}).$$

Here, G_{inj} is a constrained rewriting module; LLM rewrites r_i^- conditioned on the target factor description t_z and intensity criterion $c_{z,\ell}$, but also preserve the task-relevant content, such as final answers and named entities. This yields a controlled comparison pair $(r_i^+, \tilde{r}_{i,z,\ell}^-)$, allowing us to measure whether the injected factor changes the judge's preference.

Flip rate and effective intensity. For each comparison between r_i^+ and $\tilde{r}_{i,z,\ell}^-$, we assign $y_{i,z,\ell}^m = 1$ when m selects $\tilde{r}_{i,z,\ell}^-$, 0 for selecting r_i^+, and 0.5 for neutral cases such as ties or inconsistent decisions across orders.[1] Then, for a set of agreement instances $\mathcal{S} \subseteq \mathcal{D}_m^{\text{agr}}$, we define its flip rate under factor z at intensity level ℓ as follows:

$$\text{FR}_{m,z,\ell}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{d_i \in \mathcal{S}} y_{i,z,\ell}^m.$$

A higher flip rate implies higher susceptibility to the injected degree of bias factor, as the judge more often reverses its original preference toward the modified rejected response.

To select the most effective intensity level, we split the agreement dataset $\mathcal{D}_m^{\text{agr}}$ into a calibration split $\mathcal{D}_m^{\text{cal}}$ and a held-out test split $\mathcal{D}_m^{\text{test}}$. For each judge model m and factor z, we choose the intensity level that induces the highest flip rate on the calibration split:

$$\ell_{m,z}^* = \arg\max_{\ell \in L} \text{FR}_{m,z,\ell}(\mathcal{D}_m^{\text{cal}}).$$

We call $\ell_{m,z}^*$ the **model-specific effective intensity**. We then fix this intensity and evaluate flip rate on the held-out test split, defining the **model-factor susceptibility score** as:

$$\rho_{m,z} = \text{FR}_{m,z,\ell_{m,z}^*}(\mathcal{D}_m^{test}).$$

Thus, $\rho_{m,z}$ is the held-out flip rate of judge m for factor z under its calibration-selected intensity. Higher values indicate that the judge more often flips toward the bias-injected response. Given discovered factors $\mathcal{Z} = \{z_1, \ldots, z_K\}$, we collect the scores into a model-specific bias profile:

$$\boldsymbol{\rho}_m = [\rho_{m,z_1}, \rho_{m,z_2}, \ldots, \rho_{m,z_K}].$$

3.4. Bias Mitigation

Let us denote $y^m \in \{0, 1\}$ as the prediction of judge m on query $x \in \mathcal{D}$, *i.e.*, whether the candidate response is preferred over the reference response. Then, the win rate on dataset $\mathcal{D}$ is defined as $\text{Win}_m(\mathcal{D}) = \mathbb{E}_{x \in \mathcal{D}}[y^m]$.

Given a query x, we use a fixed scoring function G_{score} (see Section 3.3) to assign each response a scalar intensity score $\ell \in \mathcal{L}$, reflecting the strength of the target bias factor. We compute scores for the judge-preferred and human-preferred responses, denoted as ℓ_{judge} and ℓ_{human}, respectively, and define the bias intensity differential Δ as:

$$\Delta = \ell_{\text{judge}} - \ell_{\text{human}}.$$

To quantify the contribution of the bias factor to judge predictions and mitigate its effect, we model each prediction as

[1] To prevent position bias, we evaluate each comparison twice by swapping the order of the two responses.

a logistic function of three components:

$$y^m = \sigma\Big(\underbrace{\theta_m}_{\text{baseline}} + \underbrace{\phi_m \tanh(\Delta)}_{\text{bias sensitivity}} + \underbrace{\psi_m \gamma_x}_{\text{query effect}} \Big).$$

Here, θ_m captures judge m's baseline preference, ϕ_m measures sensitivity to the bias factor, γ_x represents a shared query-level effect, and ψ_m controls how strongly judge m responds to this shared signal. Given this formulation, the model is not directly identifiable due to scale ambiguity between ψ_m and γ_x. To address this, we estimate the parameters using a two-stage procedure, inspired by Length-Controlled AlpacaEval (Dubois et al., 2024) (more details can be found in Appendix F.1).

The estimated parameters enable computation of a counterfactual preference rate for judge m in the absence of the bias factor. Specifically, removing the bias term and averaging over queries yields:

$$\mathrm{BC}_m(\mathcal{D}) = \mathbb{E}_{x \in \mathcal{D}}\big[\sigma(\hat{\theta}_m + \hat{\psi}_m \hat{\gamma}_x)\big].$$

We denote BC_m as the **bias-controlled win rate**, which removes the contribution of the bias term while preserving baseline and query-level effects. The difference $\Delta_m(\mathcal{D}) = \mathrm{Win}_m(\mathcal{D}) - \mathrm{BC}_m(\mathcal{D})$ quantifies the aggregate contribution of the bias factor to judge m's observed win rate.

4. Experiment

4.1. Setups

Benchmarks. We construct the datasets for the experiments by merging a collection of five open-ended, human-annotated preference benchmarks: (1) *MT-Bench* (Zheng et al., 2023) for multi-turn chat evaluation, (2) *Multi-Pref* (Miranda et al., 2025) for multi-annotated and multi-aspect preferences, (3) *WebGPT-Comparisons* (Nakano et al., 2021) for long-form question-answering comparisons, (4) *HelpSteer2-Pref* (Wang et al., 2025a) for pairwise human preferences, and (5) *HelpSteer3* (Wang et al., 2025b) for human preference annotations over real-world LLM-use prompts. This merged benchmark allows us to analyze when LLM judges systematically deviate from human preferences across diverse open-ended tasks, after split into agreement and disagreement datasets (Section 3.1). For data with multiple human annotations, we retain only examples where at least 50% of annotators select the same response, ensuring the reliability of human preference labels.

Models. We use five representative LLMs as judge models: *GPT-4o* (OpenAI, 2024), *Claude-Haiku-4.5* (Anthropic, 2025), *Gemini-3-Flash-Preview* (Google, 2025), *Qwen-Plus* (Alibaba Cloud, 2026), and *DeepSeek-V3.2* (DeepSeek-AI, 2025). For brevity, we refer to these models as GPT, Haiku, Gemini, Qwen, and DeepSeek, respectively, throughout the paper. To mitigate position bias, we prompt each judge twice per instance by swapping the order of the two candidate responses. During data curation, we exclude tie cases but include them in flip-rate computation. Unless otherwise specified, GPT-4o is employed as the generation module $G_\bullet$. All model judgments are generated with temperature set to 0 for deterministic evaluation.

Baselines. To demonstrate the effectiveness of HUB-J, we compare against two types of baselines: (1) *pre-discovered bias factors*, collected from prior literature (Alazraki et al., 2025; Chen et al., 2024; Ye et al., 2025a; Zheng et al., 2023); and (2) *BiasScope* (Lai et al., 2026), that identifies bias factors via LLM reasoning. More details are in Appendix C.3.

Implementation details. For bias identification, we use *disagreement* dataset, where all five judge models select the same response while disagreeing with the human label, consisting of 1,830 examples split into 1,464 training and 366 test examples. For the quantification and mitigation stages, we construct an *agreement* dataset by filtering examples where all five judge models agree with the human annotation, using 100 examples for calibration and 500 examples for held-out evaluation. Additional details on implementation, models, full prompts for all LLM-based modules, bias-intensity criteria, and baseline reproduction are provided in Appendices C, H, and Table 9.

4.2. Results on Bias Identification

HUB-J's identified bias factors. For comparison, we use five representative pre-discovered bias factors: *Verbosity*, *Authority*, *Beauty*, *Sentiment*, and *Bandwagon*. We also include five factors produced by BIASSCOPE: *Availability*, *Complexity*, *Illusion of Knowledge*, *Length*, and *Risk Perception*. Full descriptions of these baseline factors are in Appendix C.3. In contrast, HUB-J extracts features directly from disagreement-case response pairs and aggregates them to construct bias factors. As shown in Table 1, the resulting bias factors are **Comprehensive List**, **Professional Tone**, **Narrative Structure**, **Implementation Focus**, and **Technical Jargon**. Qualitatively, these factors are more directly reflected in response characteristics such as structure, level of implementation detail, and use of technical terminology. To quantitatively compare these factors, we introduce two metrics evaluating their effectiveness and novelty.

Effectiveness of HUB-J's bias factors. To evaluate the effectiveness of the identified bias factors, we compute the mean Δ on disagreement test sets, where Δ is defined using the intensity scores from Section 3.4. We then average Δ across all pairs to measure the overall strength of each factor.

The results are presented in Table 2. Across all comparison groups, our method consistently achieves the strongest alignment with human–LLM disagreement patterns, reaching the highest average mean Δ (0.480), outperforming pre-discovered bias factors (0.238) and BiasScope factors

Table 1. **Top 5 bias factors of HUB-J.** The abbreviations and definitions are provided alongside and used throughout the paper for clarity.

BIAS NAME (ABBR.)	DEFINITION
Comprehensive List *(Comp. List)*	A pattern where responses that provide longer or more comprehensive lists are favored, potentially mistaking quantity for relevance.
Professional Tone *(Prof. Tone)*	The tendency to favor responses with a more professional or formal tone, interpreting them as more credible or authoritative.
Narrative Structure *(Narr. Struct)*	A bias where responses structured as narratives or stories are favored, valuing the flow and progression of the content.
Implementation Focus *(Impl. Focus)*	A bias where responses emphasizing implementation details and practical steps are preferred, potentially valuing them over broader strategic considerations.
Technical Jargon *(Tech. Jargon)*	The tendency to favor responses that use technical jargon or specialized terms, perceiving them as more expert or authoritative.

Table 2. **Comparison of bias identification results.** Mean Δ is computed per response pair the in human–LLM disagreement cases. * indicates $p < 0.01$. Full p-values are reported in Table 7

PRE-DISCOVERED		BIASSCOPE (LAI ET AL., 2026)		OURS	
BIAS (ABBR.)	MEAN Δ	BIAS (ABBR.)	MEAN Δ	BIAS (ABBR.)	MEAN Δ
Verbosity (Verb.)	+0.577*	Length bias (Leng.)	+0.732*	Comprehensive List (Comp. List)	+0.366*
Beauty (Beau.)	+0.371*	Illusion of knowledge (Illu. Know.)	+0.624*	Professional Tone (Prof. Tone)	+0.608*
Sentiment (Sent.)	+0.139*	Availability bias (Avail.)	+0.314*	Narrative Structure (Narr. Struct.)	+0.727*
Bandwagon (Band.)	+0.062	Risk perception bias (Risk Perc.)	+0.062	Implementation Focus (Impl. Focus)	+0.314*
Authority (Auth.)	+0.041	Complexity bias (Comp.)	-0.191	Technical Jargon (Tech. Jargon)	+0.387*
Average	0.238		0.308		**0.480**

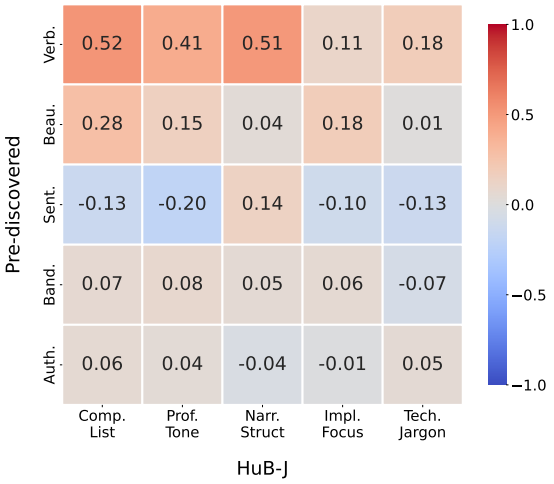


Figure 2. **Novelty analysis.** Correlation between pre-discovered factors and HUB-J factors.

(0.308). Notably, all bias factors identified by our method are statistically significant ($p < 0.01$), with mean Δ values consistently above 0.3. In contrast, several baseline factors do not reach significance, indicating weaker explanatory power for the disagreement patterns. Overall, these results show that HUB-J offers the strongest explanatory power for understanding human–LLM preference disagreements.

Novelty of the identified bias factors. Next, we evaluate the novelty of HUB-J's bias factors. We measure novelty as statistical independence from existing bias definitions, operationalized using pearson correlation. Intuitively, a novel factor should exhibit weak correlation with pre-discovered bias dimensions, with correlations close to zero indicating higher novelty due to reduced association with existing bias definitions. For this analysis, we reuse instance-level bias scores assigned by GPT-4o in Table 2 and compute correlations with pre-discovered factors.

As shown in Figure 2, *Technical Jargon*, *Implementation Focus*, and *Professional Tone* exhibit low correlations with pre-discovered bias factors, consistently remaining below 0.5 across baselines, suggesting that they capture novel dimensions not represented in existing definitions. In contrast, *Comprehensive List* and *Narrative Structure* show a positive association with *Verbosity*, a pre-discovered bias with a strong signal, indicating that HUB-J can also recover known preference dimensions. In contrast, the factors from BIASSCOPE shows higher correlations with both pre-defined bias dimensions and its own internally discovered factors, indicating substantial overlap among learned factors. This suggests that its instance-level bias identification is less effective at disentangling diverse preference dimensions, leading to reduced coverage of novel factors.

4.3. Results on Bias Quantification

Judge-specific bias profiles. With identified biases, we conduct experiments to quantify the degree of bias in each

Table 3. **Judge-specific bias profiles across bias factors.** Each cell reports the held-out flip rate (%), measuring how often the judge prefers the bias-injected response. Parenthesized values denote the calibration-selected effective intensity level ℓ^*, where larger values indicate stronger bias expression. Cell colors show the deviation from the factor-wise mean intensity score: blue indicates lower-than-average intensity, while red indicates higher-than-average intensity.

	Pre-discovered					BiasScope					Ours				
Judge	**Verb.**	**Beau.**	**Auth.**	**Band.**	**Sent.**	**Leng.**	**Comp.**	**Illu. Know.**	**Avail.**	**Risk Perc.**	**Comp. List**	**Impl. Focus**	**Prof. Tone**	**Narr. Struct**	**Tech. Jargon**
GPT	36.6 (4)	18.5 (4)	11.5 (2)	10.7 (2)	10.0 (3)	38.4 (5)	30.5 (4)	27.2 (5)	21.4 (4)	8.7 (3)	39.4 (4)	27.5 (2)	26.7 (3)	9.5 (5)	14.9 (3)
Haiku	18.6 (5)	9.4 (4)	7.9 (1)	4.6 (1)	4.3 (1)	20.2 (5)	14.5 (4)	14.3 (4)	11.2 (5)	7.4 (4)	24.8 (4)	17.4 (4)	9.4 (4)	9.9 (5)	5.3 (5)
Gemini	21.6 (4)	8.3 (4)	7.9 (2)	6.4 (2)	4.1 (3)	20.1 (5)	19.1 (5)	15.4 (5)	11.8 (4)	7.6 (5)	23.3 (5)	15.2 (2)	13.0 (4)	8.6 (5)	7.6 (4)
Qwen	26.4 (4)	8.6 (3)	6.9 (2)	5.0 (1)	4.0 (3)	25.3 (5)	18.0 (4)	16.1 (5)	12.8 (5)	7.5 (5)	25.3 (4)	17.4 (2)	13.7 (3)	6.4 (5)	7.7 (2)
DeepSeek	27.3 (4)	7.5 (5)	6.1 (3)	3.6 (2)	3.3 (1)	26.0 (5)	18.0 (5)	16.9 (5)	13.1 (4)	3.9 (3)	30.0 (4)	20.6 (4)	14.1 (3)	4.9 (5)	5.8 (2)
AVG	26.1 (4.2)	10.5 (4.0)	8.1 (2.0)	6.1 (1.6)	5.1 (2.2)	26.0 (5.0)	20.0 (4.4)	18.0 (4.8)	14.1 (4.4)	7.0 (4.0)	28.6 (4.2)	19.6 (2.8)	15.4 (3.4)	7.9 (5.0)	8.3 (3.2)

judge model. For each judge–factor pair, we evaluate five bias-injection intensity levels on the calibration split and identify the level that produces the highest flip rate as the effective intensity score ℓ^*. This selected intensity can be viewed as the bias-expression level to which the judge is most susceptible for that factor. Table 3 reports held-out flip rate on the agreement test split when the bias is injected at this selected intensity. Thus, each cell jointly presents the judge–factor susceptibility score, *i.e.*, the test flip rate, and the calibration-selected effective intensity ℓ^* in parentheses. Full calibration results are provided in Appendix E.2.

Notably, the factors discovered by HUB-J induce substantial preference flips across judge models. Compared with pre-discovered and BiasScope factors, HUB-J factors are competitive overall, with Comprehensive List achieving the highest average flip rate (28.6%). This exceeds both pre-discovered Verbosity (26.1%) and BiasScope Length (26.0%), indicating that our discovered factors capture meaningful preference shifts in LLM judges. In addition, the selected effective intensity levels vary across models and factors. For example, Haiku is most susceptible to *Technical Jargon Preference* at the strongest injection intensity ($\ell^* = 5$), whereas Qwen and DeepSeek are most affected at lower intensity levels ($\ell^* = 2$). This suggests that judges vary in their sensitivity to bias intensity: some flip when the bias is only weakly expressed, whereas others flip only when the bias is expressed more strongly. Together, these results show that judge models have distinct bias preference profiles, with some models being more vulnerable to injected bias factors and others remaining relatively robust. Figure 3 extends the quantification analysis to a broader set of LLM judges. The results show that the discovered bias factors also induce preference flips in additional judge models, while the magnitude of these effects differs across models. Some models, such as Llama-3.1-8B, Mixtral-8x7B, and Phi-4, are highly vulnerable across multiple factors, whereas models such as GPT-5, Qwen-3.5-9B, and Kimi-K2.6 remain comparatively robust. Together with the intensity-color patterns, these results support that LLM judges differ not only in vulnerability, but also in which bias factors and intensity levels most affect their decisions.

Answer-change analysis with JudgeBench. To verify whether the observed flips are driven by changes in the underlying answer quality, we further evaluate HUB-J on JudgeBench (Tan et al., 2025), whose response-pair labels are grounded in objective answer correctness rather than open-ended human preference. In Figure 4, the bars show the resulting flip rate toward the biased response, while the black labels above the bars indicate the calibration-selected effective intensity level ℓ^* for each factor–judge pair, obtained from our main experiments without further calibration. The red line reports the answer-change rate, measured by GPT-4o which determines whether the bias injection changes the underlying answer. The non-trivial flip rates under these transferred intensity levels suggest that both the discovered bias factors and their calibrated intensities generalize to a held-out evaluation benchmark. At the same time, the answer-change rate remains consistently low across factors and judge models, indicating that the observed preference shifts are primarily caused by response-level bias factors rather than changes in semantic correctness.

JudgeBench further serves as a separate benchmark for comparing the transferability of different bias-factor sets. Averaged over the five factors in each model group, our discovered factors achieve a higher flip rate than both pre-explored factors and BiasScope factors (8.4% vs. 7.0% and 6.9%, respectively; Table 8). This suggests that HUB-J's factors generalize more effectively to unseen evaluation settings. More details and results are in Appendix E.3.

4.4. Results on Bias Mitigation

To evaluate the mitigation framework, we use the training data from the quantification stage, consisting of original human–LLM agreement pairs and their bias-injected counterparts across intensity levels $\ell \in \mathcal{L}$. This setup is suitable

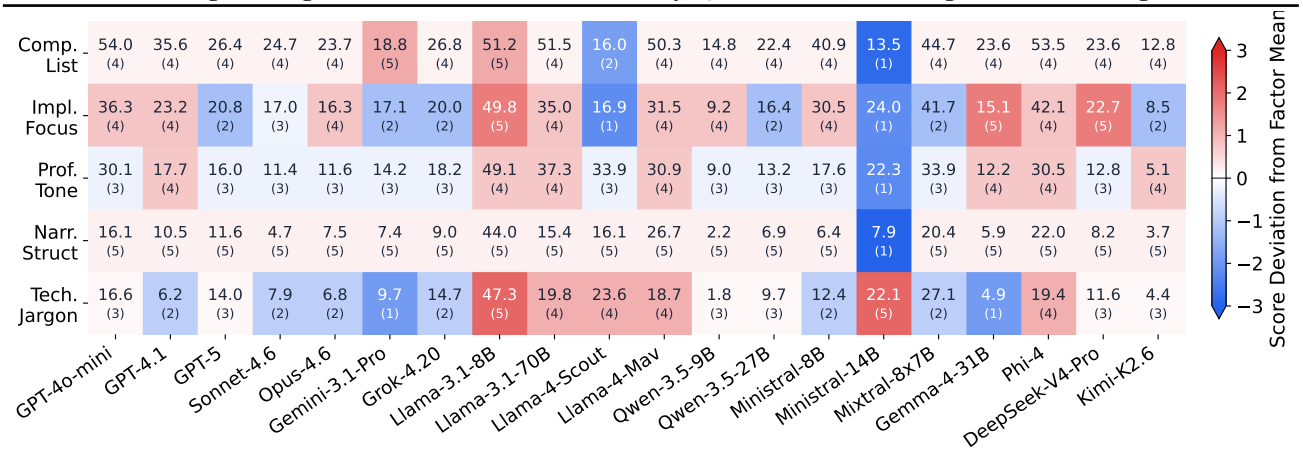


Figure 3. **Bias effects across extended 20 judge models.** Each cell reports the flip rate (%), measuring how often the judge's decision shifts toward the bias-injected response. Values in parentheses indicate the calibrated effective intensity level ℓ^*, with larger values corresponding to stronger expressions of the injected bias. Heatmap colors represent deviations of ℓ^* from the factor-wise mean, with blue indicating lower-than-average intensity and red indicating higher-than-average intensity.

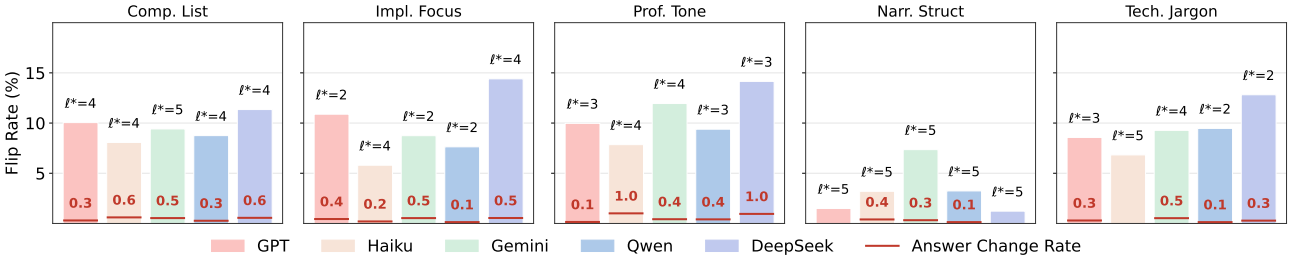


Figure 4. **Results on the JudgeBench dataset across five discovered bias factors.** For each judge model, bar heights indicate the flip rate (%). Labels above the bars denote the calibration-selected effective intensity level ℓ^* for each judge–factor pair. The red line indicates the answer change rate (%), measuring how often the response itself changes under bias injection.

Table 4. **Proportion of raw win rate attributable to bias** (%). Most susceptible judge per factor in **bold**.

Bias Factor	GPT	Haiku	Gemini	Qwen	DeepSeek	Mean
Comp. List	**56.1**	32.2	53.9	51.2	51.6	49.0
Narr. Struct	**55.7**	12.1	22.8	43.7	45.3	35.9
Prof. Tone	29.4	6.0	31.1	**36.5**	34.1	27.4
Impl. Focus	18.5	6.4	28.5	26.3	**29.2**	21.8
Tech. Jargon	**25.7**	20.0	19.9	22.6	20.7	21.8
Mean	**37.1**	15.3	31.2	36.1	36.2	31.2

as these data are naturally accumulated from the quantification stage and provide an oracle-like reference where, in the absence of bias injection, the bias-controlled win rate can be viewed as effectively neutral (BC ≈ 0), enabling clean measurement of bias-induced shifts. For each instance, we denote the intensity of the original agreement response as ℓ_{agr}. Then, we define bias exposure for each injected level as $\Delta_\ell = \ell - \ell_{\text{agr}}$, $\forall \ell \in \mathcal{L}$, similar to the intensity differential in Section 3.3. We fit models over all injected levels to capture preference shifts under varying bias exposure and evaluate debiasing at peak intensity by removing the bias term and measuring the change in flip rate.

Table 4 reports a proportion (Δ_m/Win) of each judge's win rate (*i.e.*, equal to flip rate in this case) at its effective intensity that the regression model identifies as bias-driven. On average, our method correctly recovers 31.2% of observed preferences as attributable to the imposed bias, rising to 56.1% for *Comprehensive List* which indicates that more than half of all preferences are correctly identified as bias-driven. These results confirm that our method reliably recovers bias-driven preferences.

5. Conclusion

We present HUB-J, a unified framework for discovering, quantifying, and mitigating biases in LLM-as-a-Judge systems grounded in human evaluation preferences. Unlike prior work that relies on predefined taxonomies or costly model-specific interventions, HUB-J uses human–LLM disagreement to uncover bias factors. Also, HUB-J uses agreement cases to estimate their strength via controlled perturbations, enabling a unified view of both bias presence and impact. Across experiments, HUB-J identifies both known and novel biases and reveals systematic differences in model susceptibility. For mitigation, HUB-J successfully estimates counterfactual judgments by explicitly removing bias contributions to recover unbiased preferences. Overall, HUB-J provides a comprehensive approach to understanding and addressing bias in LLM-as-a-Judge systems. Potential limitations of HUB-J are discussed in Appendix A.

References

Alazraki, L., Tan, Y.-C., Campos, J. A., Mozes, M., Rei, M., and Bartolo, M. Reverse engineering human preferences with reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=heY0zzGvYm.

Alibaba Cloud. Supported Models and Capabilities Overview. https://www.alibabacloud.com/help/en/model-studio/models, 2026. Accessed: 2026-05-04.

Anthropic. Introducing claude haiku 4.5, 2025. URL https://www.anthropic.com/news/claude-haiku-4-5.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Chen, G. H., Chen, S., Liu, Z., Jiang, F., and Wang, B. Humans or llms as the judge? a study on judgement bias. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8301–8327, 2024.

DeepSeek-AI. DeepSeek-V3.2 Release. https://api-docs.deepseek.com/news/news251201, December 2025. Accessed: 2026-05-04.

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-controlled alpacaeval: A simple way to debias automatic evaluators. In *Conference on Language Modeling*, 2024.

Google. Gemini 3 Flash Preview. https://ai.google.dev/gemini-api/docs/models/gemini-3-flash-preview, 2025. Accessed: 2026-05-04.

Jeong, H., Park, C., Hong, J., Lee, H., and Choo, J. The comparative trap: Pairwise comparisons amplifies biased preferences of llm evaluators. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 79–108, 2025.

Kim, J., Kim, T., Yoon, S., Kim, J., and Lee, D. Rpm: Reasoning-level personalization for black-box large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Kim, S., Shin, J., Cho, Y., Jang, J., Longpre, S., Lee, H., Yun, S., Shin, S., Kim, S., Thorne, J., et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Koo, H. and Kim, J. Emcee: Improving multilingual capability of llms via bridging knowledge and reasoning with extracted synthetic multilingual context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Lai, P., Ou, Z., Wang, Y., Wang, L., Yang, J., Chen, Y., and Chen, G. Biasscope: Towards automated detection of bias in llm-as-a-judge evaluation. In *International Conference on Learning Representations (ICLR)*, 2026.

Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K., Bishop, C., Hall, E., Carbune, V., Rastogi, A., et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2511–2522, 2023.

Miranda, L. J. V., Wang, Y., Elazar, Y., Kumar, S., Pyatkin, V., Brahman, F., Smith, N. A., Hajishirzi, H., and Dasigi, P. Hybrid preferences: Learning to route instances for human vs. ai feedback. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Ong, K. T.-i., Kim, N., Gwak, M., Chae, H., Kwon, T., Jo, Y., Hwang, S.-w., Lee, D., and Yeo, J. Towards lifelong dialogue agents via timeline-based memory management. pp. 8631–8661, 2025.

OpenAI. Hello gpt-4o, 2024. URL https://openai.com/index/hello-gpt-4o/.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 27730–27744, 2022.

Panickssery, A., Bowman, S. R., and Feng, S. Llm evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 68772–68802, 2024.

Raina, V., Liusie, A., and Gales, M. Is LLM-as-a-judge robust? investigating universal adversarial attacks on zero-shot LLM assessment. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Sant, A., Escolano, C., Mash, A., Fornaciari, F. D. L., and Melero, M. The power of prompts: Evaluating and mitigating gender bias in mt with llms. In *Proceedings of the 5th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, 2024.

Seo, Y., Lee, D., Kim, J., and Yeo, J. Fast and fluent diffusion language models via convolutional decoding and rejective fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=HvIRFV0J90.

Tan, S., Zhuang, S., Montgomery, K., Tang, W. Y., Cuadron, A., Wang, C., Popa, R. A., and Stoica, I. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations (ICLR)*, 2025.

Wang, P., Li, L., Chen, L., Cai, Z., Zhu, D., Lin, B., Cao, Y., Kong, L., Liu, Q., Liu, T., et al. Large language models are not fair evaluators. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024a.

Wang, Y., Yu, Z., Zeng, Z., Yang, L., Wang, C., Chen, H., Jiang, C., Xie, R., Wang, J., Xie, X., et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *International Conference on Learning Representations (ICLR)*, 2024b.

Wang, Z., Shang, J., and Zhong, R. Goal-driven explainable clustering via language descriptions. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10626–10649, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.657. URL https://aclanthology.org/2023.emnlp-main.657/.

Wang, Z., Bukharin, A., Delalleau, O., Egert, D., Shen, G., Zeng, J., Kuchaiev, O., and Dong, Y. Helpsteer2-preference: Complementing ratings with preferences. In *International Conference on Learning Representations (ICLR)*, 2025a.

Wang, Z., Zeng, J., Delalleau, O., Shin, H.-C., Soares, F., Bukharin, A., Evans, E., Dong, Y., and Kuchaiev, O. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Yang, B., Feng, L., Chen, Y., Zhang, Y., Xu, X., and Li, S. Fairjudge: An adaptive, debiased, and consistent llm-as-a-judge. *ArXiv*, abs/2602.06625, 2026. URL https://api.semanticscholar.org/CorpusID:285401808.

Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P.-Y., et al. Justice or prejudice? quantifying biases in llm-as-a-judge. In *International Conference on Learning Representations (ICLR)*, 2025a.

Ye, Z., Li, X., Li, Q., Ai, Q., Zhou, Y., Shen, W., Yan, D., and Liu, Y. Learning llm-as-a-judge for preference alignment. In *International Conference on Learning Representations (ICLR)*, 2025b.

Zeng, Z., Yu, J., Gao, T., Meng, Y., Goyal, T., and Chen, D. Evaluating large language models at evaluating instruction following. In *International Conference on Learning Representations (ICLR)*, 2024.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Appendix

A. Limitations

HUB-J introduces additional computational cost because it requires LLM calls for bias discovery, bias injection, and judge-based quantification. Once identified, they can be reused across benchmarks and judge models, reducing the impact of the initial discovery cost. Our JudgeBench transfer results in Figure 4 and Table 8 further show that these factors generalize beyond the original discovery setting. Our method also requires human preference labels to define disagreement and agreement between humans and LLM judges. This may limit its use in settings where such labels are unavailable. However, human preferences provide the natural reference point for evaluating LLM-as-a-Judge systems, whose goal is to approximate human evaluation. By grounding bias discovery in human–LLM judgment gaps, our framework identifies biases that are directly relevant to human-aligned evaluation, rather than relying solely on predefined or model-generated bias factors.

B. Algorithmic Details of HUB-J

We present the algorithmic details for each stage of HUB-J, shown sequentially in Algorithms1 (bias identification), 2 (bias quantification), and 3 (bias mitigation).

Algorithm 1 Bias Factor Identification

Require: $\mathcal{D}^{\text{dis}}_{\mathcal{M}}$, target factors K, thresholds τ, ρ, max iters T, weights α, λ_u
Ensure: Bias factor set $\mathcal{Z}$
Extract $\mathcal{F}_i \leftarrow G_{\text{extract}}(d_i)$ for each $d_i \in \mathcal{D}^{\text{dis}}_{\mathcal{M}}$ and set $\mathcal{F} \leftarrow \bigcup_i \mathcal{F}_i$
Initialize $\mathcal{Z} \leftarrow \emptyset$, $\tilde{\mathcal{Z}} \leftarrow \emptyset$, $\mathcal{F}^{(0)} \leftarrow \mathcal{F}$
for $t = 1, \ldots, T$ **do**
    Cluster $\mathcal{F}^{(t-1)}$ into M groups via K-means and pick the feature closest to each centroid as representatives $\mathcal{R}$
    Generate candidate factors $\mathcal{Z}^{\text{cand}}$ and descriptions $\{t_z\}$ via $G_{\text{construct}}(\mathcal{R})$
    Add each $z \in \mathcal{Z}^{\text{cand}}$ to $\tilde{\mathcal{Z}}$ if its cosine similarity to every $z' \in \tilde{\mathcal{Z}}$ is below τ
    Assign features via $a \leftarrow G_{\text{assign}}(\mathcal{F}, \tilde{\mathcal{Z}})$, with $a(f) \in \tilde{\mathcal{Z}} \cup \{\perp\}$
    Set $\mathcal{C}(z) \leftarrow \{f : a(f) = z\}$ for each $z \in \tilde{\mathcal{Z}}$ and $\mathcal{F}_{\text{unc}} \leftarrow \{f : a(f) = \perp\}$
    while $|\mathcal{Z}| < K$ and $\tilde{\mathcal{Z}} \setminus \mathcal{Z} \neq \emptyset$ **do**
        Update $\mathcal{U}(\mathcal{Z}) \leftarrow \bigcup_{z \in \mathcal{Z}} \mathcal{C}(z)$
        Select $z^* \leftarrow \arg\max_{z \in \tilde{\mathcal{Z}} \setminus \mathcal{Z}} \alpha |\mathcal{C}(z) \setminus \mathcal{U}(\mathcal{Z})| - \lambda_u |\mathcal{F}_{\text{unc}}|$, then $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{z^*\}$
    end while
    if $|\mathcal{U}(\mathcal{Z})|/|\mathcal{F}| \geq \rho$ **then**
        break
    end if
    $\mathcal{F}^{(t)} \leftarrow \mathcal{F}_{\text{unc}}$
end for
return $\mathcal{Z}$ =0

C. Implementation Details

C.1. Dataset Details

Human preference filtering. For datasets with multiple human annotations, we first apply a quality-control filtering step to ensure that each example has a reliable human preference label. Specifically, we retain only examples for which at least 50% of the annotators select the same response, and use this response as the human preference label. Examples that do not satisfy this criterion are discarded.

Disagreement dataset. To identify bias factors, we construct a *disagreement* dataset consisting of examples where the judge models exhibit unanimous agreement with one another but disagree with the human preference label. For each filtered example, we obtain preference judgments from five judge models: GPT, Haiku, Gemini, Qwen, and DeepSeek. An example is included in the disagreement dataset if all five judge models select the same response, while their shared prediction differs from the human preference label. For instance, if the human majority label is Response A but all five judge models select Response B, the example is included in the disagreement dataset.

Algorithm 2 Bias Quantification

Require: Judge model m, agreement set $\mathcal{D}_m^{agr}$, bias factors $\mathcal{Z}$, G_{crit}, G_{inj}
Ensure: Bias profile ρ_m
Split $\mathcal{D}_m^{agr}$ into $\mathcal{D}_m^{cal}$ and $\mathcal{D}_m^{test}$
for each $(z, t_z) \in \mathcal{Z}$ **do**
  $\mathcal{C}_z \leftarrow G_{\text{crit}}(z, t_z) = \{c_{z,\ell}\}_{\ell=1}^{5}$
  for $\ell = 1$ to 5 **do**
    $\text{FR}_{m,z,\ell}^{cal} \leftarrow \text{FlipRate}(m, \mathcal{D}_m^{cal}, z, t_z, c_{z,\ell})$
  end for
  $\ell_{m,z}^{*} \leftarrow \arg\max_{\ell \in \{1,\ldots,5\}} \text{FR}_{m,z,\ell}^{cal}$
  $\rho_{m,z} \leftarrow \text{FlipRate}(m, \mathcal{D}_m^{test}, z, t_z, c_{z,\ell_{m,z}^{*}})$
end for
$\rho_m \leftarrow [\rho_{m,z_1}, \rho_{m,z_2}, \ldots, \rho_{m,z_K}]$
return ρ_m
=0

Algorithm 3 Bias-Controlled Win Rate Estimation

Require: Dataset $\mathcal{D}$, judges $\{m\}$, predictions $\{y^m\}_{x \in \mathcal{D}}$, scoring function G_{score}
Ensure: $\text{BC}_m(\mathcal{D})$ and $\Delta_m(\mathcal{D})$ for each judge m
for each $x \in \mathcal{D}$ **do**
  $\Delta_x \leftarrow \tanh\big(G_{\text{score}}(\text{judge-pref.}) - G_{\text{score}}(\text{human-pref.})\big)$
end for
Stage I. Jointly fit $y^m = \sigma(\theta_m + \phi_m \Delta_x + \psi_m \gamma_x)$ over all (m, x) with identifiability anchor $\psi_1 = 1$; retain $\{\hat{\gamma}_x\}$.
Stage II. For each m, with $\hat{\gamma}_x$ fixed,

$$(\hat{\theta}_m, \hat{\phi}_m, \hat{\psi}_m) \leftarrow \arg\max \ \mathbb{E}_{x \in \mathcal{D}}\big[\log \sigma\big((2y^m - 1)(\theta + \phi\, \Delta_x + \psi\, \hat{\gamma}_x)\big)\big].$$

for each judge m **do**
  $\text{Win}_m \leftarrow \mathbb{E}_{x \in \mathcal{D}}[y^m]$
  $\text{BC}_m \leftarrow \mathbb{E}_{x \in \mathcal{D}}\big[\sigma(\hat{\theta}_m + \hat{\psi}_m\, \hat{\gamma}_x)\big]$ {$\tanh(0) = 0$ drops $\hat{\phi}_m \Delta_x$}
  $\Delta_m \leftarrow \text{Win}_m - \text{BC}_m$
end for
return $\{\text{BC}_m, \Delta_m\}_m$ =0

This filtering procedure yields 1,830 disagreement examples in total. We split these examples into 1,464 training samples and 366 test samples. The training split is used for bias factor discovery, while the test split is used as a validation set to verify the identified factors, for example by computing the mean Δ associated with each factor.

Agreement dataset. For the quantification and mitigation stages, we additionally construct an *agreement* dataset. In contrast to the disagreement dataset, this dataset contains examples where the human annotators and all five judge models select the same response. For example, if the human preference label is Response A and all five judge models also select Response A, the example is included in the agreement dataset.

From this agreement dataset, we sample 100 examples for calibration and 500 examples for held-out evaluation. The calibration set is used to calibrate the bias quantification and mitigation procedures, while the held-out evaluation set is used to assess their effectiveness on examples not used during calibration.

Hyperparameter setting. During the bias discovery stage, we set the cosine similarity threshold to $\tau = 0.7$ and select the top-5 factors, *i.e.*, Top-$K = 5$. These selected factors are then used throughout all subsequent experiments.

C.2. Model Details

Main experiments. We use five representative LLM judges:

- GPT-4o (`gpt-4o-2024-08-06`)
- Claude-Haiku-4.5 (`claude-haiku-4-5-20251001`)
- Gemini-Flash-3 (`gemini-3-flash-preview-202512`)
- Qwen-Plus (`qwen/qwen-plus`)
- DeepSeek-V3.2 (`deepseek/deepseek-v3.2`)

Extended analysis. For the model-extension study, we additionally evaluate the following 20 LLM judges, grouped by model family:

- **OpenAI:** GPT-4o-mini (`gpt-4o-mini-2024-07-18`), GPT-4.1 (`gpt-4.1-2025-04-14`), GPT-5 (`gpt-5-2025-08-07`)
- **Anthropic:** Claude-Sonnet-4.6 (`claude-sonnet-4-6`), Claude-Opus-4.6 (`claude-opus-4-6`)
- **Google:** Gemini-3.1-Pro (`gemini-3.1-pro-preview-202602`), Gemma-4-31B (`google/gemma-4-31b-it`)
- **xAI:** Grok-4.20 (`x-ai/grok-4.20`)
- **Meta:** Llama-3.1-8B (`meta-llama/llama-3.1-8b-instruct`), Llama-3.1-70B (`meta-llama/llama-3.1-70b-instruct`), Llama-4-Scout (`meta-llama/llama-4-scout`), Llama-4-Maverick (`meta-llama/llama-4-maverick`)
- **Alibaba:** Qwen-3.5-9B (`qwen/qwen3.5-9b`), Qwen-3.5-27B (`qwen/qwen3.5-27b`)
- **Mistral:** Ministral-8B (`mistralai/ministral-8b-2512`), Ministral-14B (`mistralai/ministral-14b-2512`), Mixtral-8x7B (`mistralai/mixtral-8x7b-instruct`)
- **Microsoft:** Phi-4 (`microsoft/phi-4`)
- **DeepSeek:** DeepSeek-V4-Pro (`deepseek/deepseek-v4-pro`)
- **Moonshot:** Kimi-K2.6 (`moonshotai/kimi-k2.6`)

GPT-family models are accessed through the OpenAI API, Claude-family models through the Anthropic API, and Gemini-family models through the Google API. All remaining models are accessed through OpenRouter[2].

C.3. Baseline Details and Factors

To evaluate the effectiveness of HUB-J, we compare against two main baselines as follows:

Pre-discovered bias factors. We incorporate bias factors identified in prior literature (Alazraki et al., 2025; Chen et al., 2024; Ye et al., 2025a; Zheng et al., 2023), which are presented in Table 5 along with their definitions. We also adopt the corresponding definitions from their original sources, and use them for comparison with our method.

BIASSCOPE. BIASSCOPE (Lai et al., 2026) is an automatic bias discovery pipeline that leverages LLM reasoning to identify bias factors. It first generates judges' self-explanations for their decisions on each pairwise comparison and uses a separate teacher model to analyze them and propose instance-level bias factors. These candidates are then validated via a flip-rate evaluation. Specifically, bias is injected into rejected responses, and the model's preference reversal is measured against perturbed and original versions.

To adapt this mechanism as a baseline in our work, we generate judges' reasoning on the human–LLM disagreement dataset and use GPT-4o to analyze them and propose candidate bias factors. We then evaluate these factors on the quantification dataset (i.e., a subset of agreement cases) by injecting each bias into rejected responses and constructing a neutralized counterpart via prompting. Then, judges re-evaluate the paired responses, and we compute the flip rate. We pick the top-5 factors with the highest flip rates to compare with HUB-J's factors, and their descriptions are provided in Table 6.

[2] https://openrouter.ai/

Table 5. **Pre-discovered bias factors.** We present the bias factors collected from prior literature.

BIAS NAME (ABBR.)	DEFINITION
Verbosity *(Verb.)*	LLM judge favors longer, verbose responses, even if they are not as clear, high-quality, or accurate as shorter alternatives.
Beauty *(Beau.)*	The inclination that judges tend to prefer visually appealing content, regardless of its actual validity.
Authority *(Auth.)*	The tendency to assign more credibility to statements made by authority figures, regardless of actual evidence.
Bandwagon *(Band.)*	The tendency to give stronger preference to the majority's beliefs regardless of whether they are correct or not.
Sentiment *(Sent.)*	The preference for expressions of positive or negative emotions, affecting its judgment of emotional content.

Table 6. **BIASSCOPE bias factors.** The following factors are identified using an automatic bias discovery approach based on the judge model's self-explanations and LLM reasoning.

BIAS NAME (ABBR.)	DEFINITION
Length Bias *(Leng.)*	Length bias refers to the tendency of large language models (LLMs) to prefer longer generated outputs when evaluating text quality, while disregarding the actual content quality or relevance.
Complexity Bias *(Comp.)*	Complexity bias is the tendency to give undue preference to more complex explanations or solutions over simpler ones, even when the simpler option is sufficient or more appropriate.
Illusion of Knowledge *(Illu. Know.)*	The illusion of knowledge bias occurs when an individual believes they understand a topic more deeply than they actually do, often due to superficial or incomplete information.
Availability Bias *(Avail.)*	Availability bias is a cognitive bias that occurs when people rely on immediate examples that come to mind when evaluating a specific topic, concept, method, or decision.
Risk Perception Bias *(Risk Perc.)*	Risk perception bias refers to the tendency to underestimate or overestimate the risks associated with certain activities or situations.

D. Additional Results on Bias Identification

D.1. Statistical Significance of Bias Identification Results

Here we present the full version of Table 2 with p-values. Our method yields statistically significant effects across all identified factors, whereas several baseline factors fail to reach statistical significance.

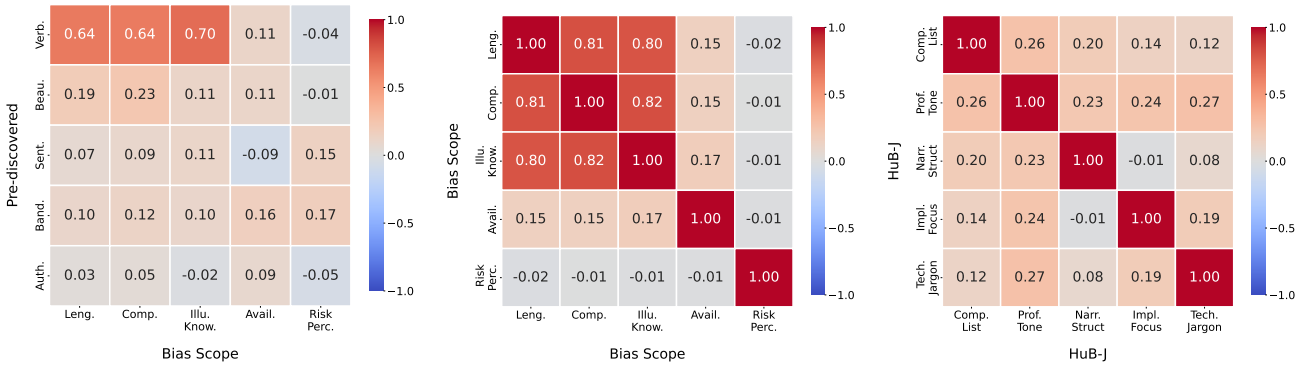
D.2. Additional Novelty Analysis of Discovered Biases

To further explore the novelty of HUB-J, we compare our discovered factors with those from BIASSCOPE by computing Pearson correlations between BIASSCOPE factors and predefined bias factors. The collective results are presented in Figure 5. Analytically, three out of five BIASSCOPE factors exhibit high correlation with *Verbosity*, indicating a strong dependency on a single dominant bias axis. We further analyze the internal correlations among BIASSCOPE factors, which again reveal high inter-factor correlations. This suggests that BIASSCOPE has limitations in producing diverse bias factors, as the generated factors are not sufficiently independent from each other.

In contrast, the internal correlation analysis of HUB-J shows relatively low correlations among the discovered factors. This indicates that our approach produces more diverse and less redundant factors, capturing a broader range of bias dimensions.

Table 7. **Bias-factor validation results** reporting each factor, its abbreviation, mean preference shift Δ, and corresponding p-value.

PRE-DISCOVERED		BIASSCOPE (LAI ET AL., 2026)		OURS	
BIAS (ABBR.)	MEAN Δ (P)	BIAS (ABBR.)	MEAN Δ (P)	BIAS (ABBR.)	MEAN Δ (P)
Verbosity (Verb.)	+0.577 (< .001)	Length bias (Leng.)	+0.732 (< .001)	Comprehensive List (Comp. List)	+0.366 (< .001)
Beauty (Beau.)	+0.371 (< .001)	Illusion of knowledge (Illu. Know.)	+0.624 (< .001)	Professional Tone (Prof. Tone)	+0.608 (< .001)
Sentiment (Sent.)	+0.139 (0.0019)	Availability bias (Avail.)	+0.314 (< .001)	Narrative Structure (Narr. Struct.)	+0.727 (< .001)
Bandwagon (Band.)	+0.062 (0.3252)	Risk perception bias (Risk Perc.)	+0.062 (0.0703)	Implementation Focus (Impl. Focus)	+0.314 (< .001)
Authority (Auth.)	+0.041 (0.2492)	Complexity bias (Comp.)	-0.191 (0.2129)	Technical Jargon (Tech. Jargon)	+0.387 (< .001)
Average	0.238		0.308		**0.480**



(a) BIASSCOPE External *(b)* BIASSCOPE Internal *(c)* HUB-J Internal

Figure 5. **Correlation analysis** across different bias discovery frameworks.

D.3. Model Dependency in Bias Factor Identification

In this section, we investigate whether using different LLMs for feature extraction and factorization can still yield strong bias factors. To this end, we run the bias identification pipeline using Claude-Haiku-4.5 and evaluate the resulting mean Δ. For a fair comparison with GPT-4o, we use the same evaluation criteria and scoring procedure. Results are reported in Figure 9.

The results show that, similar to GPT-4o, factors discovered using Claude-Haiku-4.5 also achieve strong mean Δ values with high statistical significance. Their novelty is further supported by relatively low correlations with predefined factors, except for Comprehensive Coverage, which is also correlated with Verbosity. The internal correlation among Claude-based factors remains low (max 0.52), indicating that the identified factors are still diverse.

However, when we compute Pearson correlations between factors discovered by Claude-Haiku-4.5 and those discovered by GPT-4o, we observe only limited overlap. This suggests that different LLMs may leverage distinct internal priors or knowledge structures when identifying bias factors, enabling the discovery of different bias perspectives. At the same time, the presence of partial overlap also highlights a limitation of purely LLM-driven bias discovery, where model-specific biases may influence the resulting factor space.

E. Additional Results on Bias Quantification

E.1. Likert-Scale Criteria for Bias Intensity

For each discovered bias factor, we define five ordered intensity levels on a 1–5 Likert scale. A larger ℓ indicates a stronger expression of the corresponding bias factor. Table 9 summarizes the criteria used for bias injection.

E.2. Additional Results on Calibration Datasets

For the calibration analysis, we use 100 samples whose human annotations agree with the outputs of all five models: GPT, Haiku, Gemini, Qwen, and DeepSeek. For each bias factor, we inject bias at target 1–5 Likert scale to evaluate the resulting flip rates using the same five judge models. Figures 10, 11, and 12 report the results for the pre-discovered, BiasScope, and HUB-J bias factors, respectively.

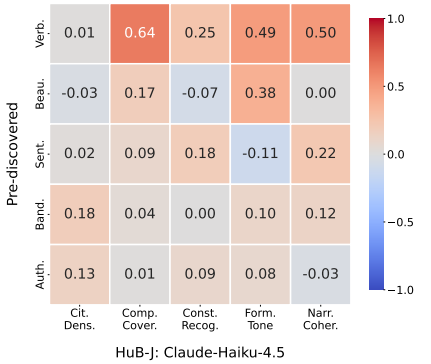


Figure 6. Haiku External

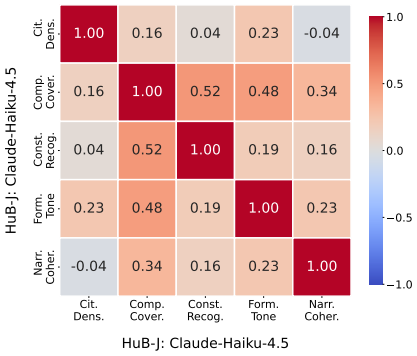


Figure 7. Haiku Internal

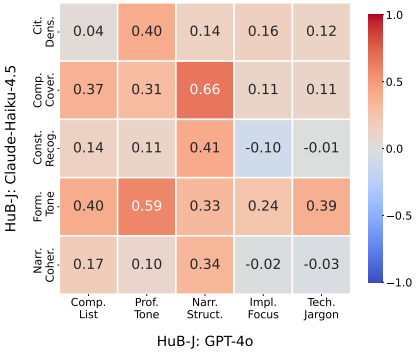


Figure 8. Haiku vs. GPT-4o

Figure 9. **Various correlation analysis** for Claude-Haiku-4.5.

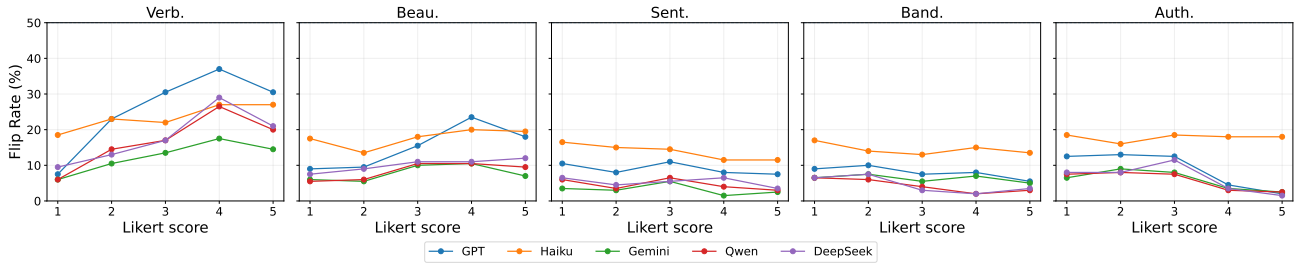


Figure 10. **Flip rates on the calibration set for Pre-discovered bias factors.**

Across the three sets of bias factors, we observe that flip rates are highly model-dependent. Even under the same bias factor and Likert level, different judge models respond with substantially different flip rates, indicating that the injected bias does not affect all models uniformly. Moreover, the relationship between the target Likert level and the flip rate is not always monotonic. While some factors produce larger flip rates at higher intensity levels, others plateau or peak at intermediate levels. These trends suggest that each judge model has its own vulnerable bias factors and intensity ranges, motivating model-specific calibration when quantifying bias effects.

E.3. Additional Experiments on JudgeBench

Setting. JudgeBench (Tan et al., 2025) is a benchmark for evaluating LLM-based judges on objective correctness over challenging response pairs. It consists of 620 queries[3], where each response pair contains two responses generated by the same source model, either Claude-3.5-Sonnet or GPT-4o. Each pair is annotated according to whether the responses are consistent with the ground-truth label. We use JudgeBench as an unseen benchmark to test whether the bias factors identified in our main experiments transfer beyond the original discovery setting. For each bias factor, we reuse the bias intensity score ℓ^* obtained from the main experiment.

Experiment. We apply the same perturbation protocol to JudgeBench and compare three groups of bias factors: pre-discovered factors, BiasScope factors, and our discovered factors. Table 8 reports the flip rate and the corresponding bias intensity score ℓ^* for each judge model and factor. Overall, HuB-J factors achieve the highest average flip rate on JudgeBench. Averaging over the five factors in each group, our factors obtain an average flip rate of 8.4%, compared to 7.0% for the pre-discovered factors and 6.9% for BiasScope. This suggests that the proposed factors are not specific to the original discovery data, but generalize well to an unseen judge-evaluation benchmark.

The improvement is especially clear for several individual factors. Among all factors, our *Professional Tone* factor yields the highest average flip rate, reaching 10.1%. Other discovered factors, including *Comprehensive List*, *Implementation Focus*, and *Technical Jargon*, also achieve strong average flip rates of 9.4%. These results are competitive with or higher than the strongest pre-discovered factor, *Verbosity* at 9.6%, and the strongest BiasScope factors, such as *Length* at 8.6%

[3]https://huggingface.co/datasets/ScalerLab/JudgeBench

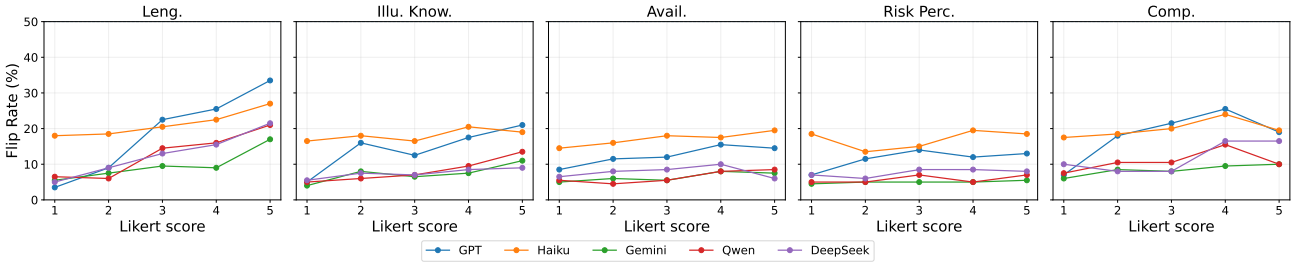


Figure 11. **Flip rates on the calibration set for BiasScope bias factors.**

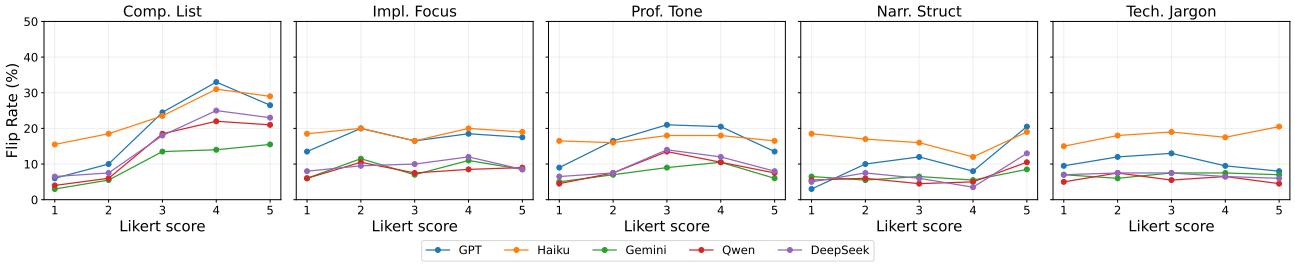


Figure 12. **Flip rates on the calibration set for HUB-J bias factors.**

and *Complexity* at 8.5%. Across judge models, our factor group produces the highest average flip rate for GPT, Qwen, and DeepSeek, while remaining competitive for Haiku and Gemini.

Importantly, this gain is not simply due to using the largest perturbation intensities. The average ℓ^* of our factors is 3.7, which is lower than that of BiasScope (4.5), despite achieving a higher average flip rate. Although our factors use stronger perturbations on average than the pre-discovered factors (2.8), they also outperform BiasScope, whose perturbation intensities are higher. This suggests that the higher flip rates are not merely an artifact of more aggressive perturbations, but reflect bias dimensions that transfer effectively to JudgeBench.

E.4. Ablation Studies with Different LLMs

To examine whether our results depend on a specific rewriting model, we conduct additional ablation studies on JudgeBench using different rewriter models, including Haiku, Gemini, Qwen, and DeepSeek. Figures 13, 14, 15 and 16 show that the discovered bias factors continue to induce non-trivial preference flips across judge models even when the rewriter is changed. This suggests that the observed bias effects are not specific to a single rewriter model.

At the same time, the rewriter model affects how well the underlying answer content is preserved during bias injection. Compared with the alternative rewriters, GPT-4o, used in our main experiments in Figure 4, produces a lower answer-change rate while still yielding clear preference flips. We therefore use GPT as the main rewriter model, as it better preserves answer quality and allows us to attribute the observed flips more directly to response-level bias factors rather than changes in semantic correctness.

F. Additional Results on Bias Mitigation

F.1. Two-Stage Estimation Procedure

The model

$$y^m = \sigma\big(\theta_m + \phi_m \tanh(\Delta) + \psi_m \gamma_x\big)$$

is not directly identifiable, because the parameters ψ_m and γ_x appear only as the product $\psi_m \gamma_x$. Intuitively, the data reveals how much the shared query signal contributes to each prediction, but not whether that contribution comes from large query effects γ_x that judges weight weakly, or from small query effects that judges weight strongly. Formally, for any nonzero constant c, the substitution

$$\gamma_x \mapsto c\,\gamma_x, \qquad \psi_m \mapsto \psi_m / c$$

Table 8. **Flip-rate results on JudgeBench with bias intensity scores (ℓ^*) from main experiment.**

	Pre-discovered					BiasScope					Ours				
Judge	Verb.	Beau.	Auth.	Band.	Sent.	Leng.	Comp.	Illu. Know.	Avail.	Risk Perc.	Comp. List	Impl. Focus	Prof. Tone	Narr. Struct	Tech. Jargon
GPT	9.1 (4)	6.9 (4)	3.1 (2)	2.5 (2)	3.1 (3)	8.2 (5)	7.8 (4)	7.9 (5)	3.0 (4)	1.3 (3)	9.7 (4)	10.4 (2)	9.8 (3)	1.5 (5)	8.3 (3)
Haiku	7.8 (5)	7.7 (4)	5.8 (1)	5.6 (1)	5.8 (1)	5.8 (5)	6.8 (4)	8.0 (4)	5.4 (5)	5.1 (4)	7.8 (4)	5.8 (4)	7.4 (4)	2.2 (5)	7.1 (5)
Gemini	10.9 (4)	7.9 (4)	7.6 (2)	7.5 (2)	8.3 (3)	10.8 (5)	10.6 (5)	10.4 (5)	9.8 (4)	6.7 (5)	8.3 (5)	8.5 (2)	11.1 (4)	8.1 (5)	9.3 (4)
Qwen	8.6 (4)	7.1 (3)	5.0 (2)	6.6 (1)	3.6 (3)	8.2 (5)	6.6 (4)	6.2 (5)	3.2 (5)	2.6 (5)	8.5 (4)	7.5 (2)	9.0 (3)	3.1 (5)	9.3 (2)
DeepSeek	11.6 (4)	7.4 (5)	5.8 (3)	5.4 (2)	14.7 (1)	9.9 (5)	10.5 (5)	7.8 (5)	4.5 (4)	4.7 (3)	12.9 (4)	14.6 (4)	13.1 (3)	2.6 (5)	12.8 (2)
AVG	9.6 (4.2)	7.4 (4)	5.5 (2)	5.5 (1.6)	7.1 (2.2)	8.6 (5)	8.5 (4.4)	8.1 (4.8)	5.2 (4.4)	4.1 (4)	9.4 (4.2)	9.4 (2.8)	10.1 (3.4)	3.5 (5)	9.4 (3.2)

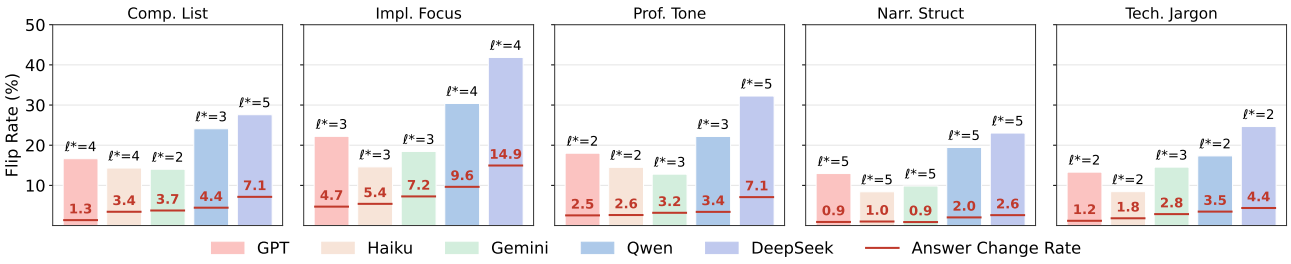


Figure 13. **Results on the JudgeBench dataset using Haiku as rewriter model**

leaves the product $\psi_m \gamma_x$ unchanged, and therefore yields identical predicted probabilities and likelihood. Infinitely many parameter values are thus consistent with any dataset, and ψ_m and γ_x cannot be jointly recovered without an additional constraint. We resolve this by estimating the query-level effects and the per-judge parameters in two stages, following the strategy of Length-Controlled AlpacaEval (Dubois et al., 2024).

Stage 1: Estimating query-level effects. We fix $\psi_m = 1$ for all judges m, removing the multiplicative degree of freedom, and fit

$$y^m = \sigma\big(\theta_m + \phi_m \tanh(\Delta) + \gamma_x\big)$$

jointly over all judges m and queries $x \in \mathcal{D}$ by maximum likelihood. Joint estimation across judges is essential: γ_x is identified as the component of variation in y^m that is *shared across judges on the same query*, and a single judge alone cannot separate per-query effects from per-instance noise. To eliminate the remaining additive freedom between $\{\theta_m\}$ and $\{\gamma_x\}$, we additionally pin $\gamma_{x_0} = 0$ for one reference query $x_0 \in \mathcal{D}$. Stage 1 thus produces estimates $\{\hat{\gamma}_x\}_{x\in\mathcal{D}}$, while the per-judge parameters obtained in this pass are intermediate and are discarded in favor of the Stage 2 estimates.

We then re-center the recovered query effects to have zero mean across $\mathcal{D}$,

$$\hat{\gamma}_x \;\leftarrow\; \hat{\gamma}_x - \mathbb{E}_{x\in\mathcal{D}}[\hat{\gamma}_x],$$

which fixes the absolute level of the query effects. After re-centering, θ_m in Stage 2 is interpretable as judge m's log-odds at the average query, and $\hat{\gamma}_x$ as the deviation of query x from that average.

Stage 2: Estimating per-judge parameters. Treating the centered $\{\hat{\gamma}_x\}_{x\in\mathcal{D}}$ from Stage 1 as a known covariate, we refit

$$y^m = \sigma\big(\theta_m + \phi_m \tanh(\Delta) + \psi_m \hat{\gamma}_x\big)$$

separately for each judge m by maximum likelihood. Because $\hat{\gamma}_x$ is now fixed, ψ_m is identified for each judge and captures judge m's individual weighting on the shared query signal, rather than being absorbed into the global scale set in Stage 1. This yields the final estimates $(\hat{\theta}_m, \hat{\phi}_m, \hat{\psi}_m)$ together with valid standard errors from the per-judge logistic regression. The full set $\{\hat{\theta}_m, \hat{\phi}_m, \hat{\psi}_m\}_m$ and $\{\hat{\gamma}_x\}_{x\in\mathcal{D}}$ is then used to compute the bias-controlled win rate $\mathrm{BC}_m(\mathcal{D})$.

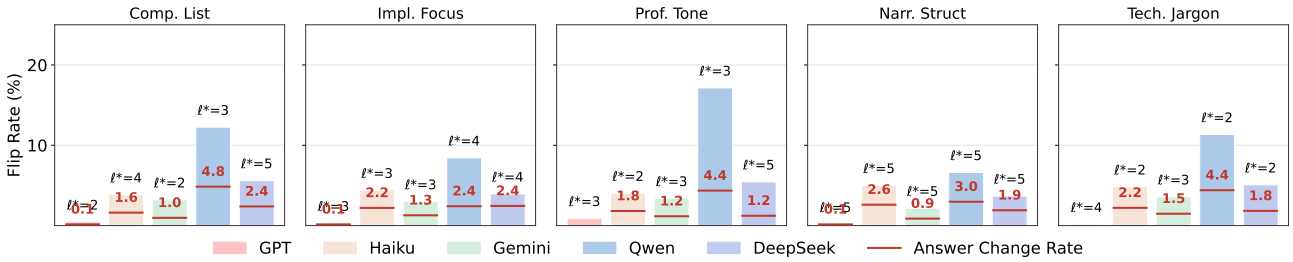


Figure 14. **Results on the JudgeBench dataset using Gemini as rewriter model**

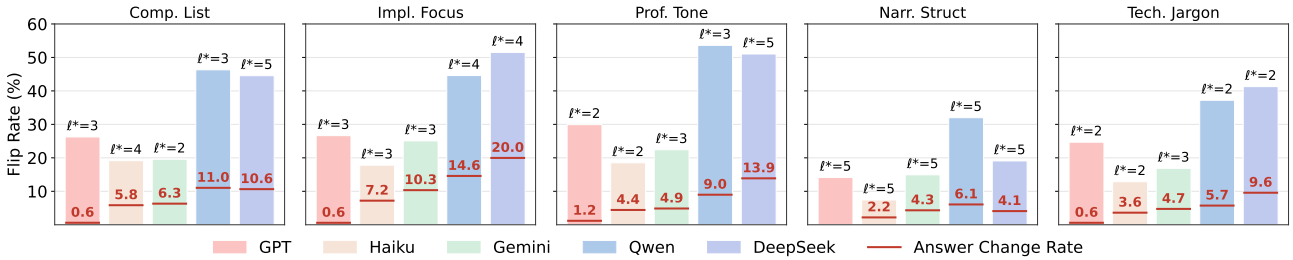


Figure 15. **Results on the JudgeBench dataset using Qwen as rewriter model**

F.2. Full Bias Contribution Results at Peak Intensity

Table 10 reports the full raw win rate, bias-controlled win rate (BC), and bias contribution Δ at peak intensity for each (factor, judge) pair. All pairs yield $\Delta > 0$, confirming that every bias factor produces a measurable preference shift in every judge.

Table 10. **Bias contribution at peak intensity.** For each (factor, judge) pair we report the raw win rate, the bias-controlled win rate (BC), and the bias contribution Δ = raw − BC, all in percentage points.

Bias factor	Metric	DeepSeek	Gemini	Haiku	GPT	Qwen	Mean
Comp. List	Raw	25.00	15.00	31.00	43.75	22.00	27.35
	BC	12.11	6.91	21.03	19.19	10.75	14.00
	Δ	**12.89**	**8.09**	**9.97**	**24.56**	**11.25**	**13.35**
Impl. Focus	Raw	11.75	11.75	19.75	17.25	10.50	14.20
	BC	8.32	8.40	18.49	14.06	7.74	11.40
	Δ	**3.43**	**3.35**	**1.26**	**3.19**	**2.76**	**2.80**
Narr. Struct	Raw	13.00	8.25	19.00	25.50	10.50	15.25
	BC	7.11	6.37	16.70	11.29	5.92	9.48
	Δ	**5.89**	**1.88**	**2.30**	**14.21**	**4.58**	**5.77**
Prof. Tone	Raw	14.25	10.50	18.00	26.50	13.50	16.55
	BC	9.39	7.23	16.93	18.71	8.57	12.17
	Δ	**4.86**	**3.27**	**1.07**	**7.79**	**4.93**	**4.38**
Tech. Jargon	Raw	7.75	6.50	20.50	12.00	7.50	10.85
	BC	6.14	5.21	16.39	8.91	5.81	8.49
	Δ	**1.61**	**1.29**	**4.11**	**3.09**	**1.69**	**2.36**

G. Usage of AI Assistants

We used AI-based writing assistants in a limited manner during the preparation of this paper, mainly to improve grammar, sentence clarity, and readability. Their use was restricted to editorial support and did not contribute to the formulation of the research questions, the design of the methodology, the execution of experiments, or the interpretation of results. All technical ideas, analyses, and conclusions were developed and verified by the authors. Therefore, the use of AI assistance

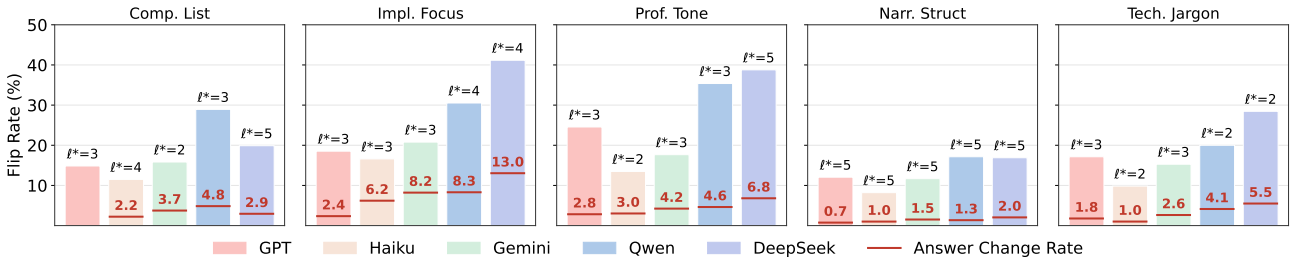


Figure 16. **Results on the JudgeBench dataset using DeepSeek as rewriter model**

does not affect the originality, novelty, or intellectual contributions of this work.

H. Designed Prompts for Experiment

Score Assignment to Responses

```
SYS = """You are a helpful assistant that outputs JSON."""

USER = f"""
You are an objective evaluator grading an AI response.
Score the following response on a scale of 1 to 5 based STRICTLY on the criteria below.
You must select EXACTLY ONE score.

Do not explain your reasoning.
Output ONLY the JSON object.

BIAS FACTOR: {preference_name}

BIAS DEFINITION: {bias_definition}

CRITERIA:
{criteria_json}

USER QUESTION:
{query}

RESPONSE TO EVALUATE:
{response_text}

Return STRICTLY a JSON object with a single integer value:
{{
  "score": <integer 1–5>
}}
```

Feature Extraction Prompt

```
You are given a case where a human evaluator and an LLM judge evaluated two responses to a query and disagreed on which
      response is better.

The human label is treated as the ground truth. The LLM judge's disagreement therefore reflects a potential systematic bias.

Your task is to extract interpretable features that explain why the LLM judge may have been biased toward its choice, causing it to
      deviate from the human ground truth.

---

Inputs:
- Original user query
- r1 (Response 1) and r2 (Response 2)
- human_label: what the human preferred ("r1", "r2", or "tie")
- model_chose: what the LLM judge chose ("r1", "r2", or "tie")

These always disagree. Your goal is to find features in the response pairs that may have unfairly swayed the LLM judge away
      from the human's judgment.



IMPORTANT:
- Focus on features that explain the gap between human judgment and LLM judgment, not just general quality differences.
- Also consider whether the judge may prefer answers that sound more expert, more scientific, more emotional, more supportive,
      more professional, more teacher-like, more friend-like, more neutral, more experienced, or more number-heavy.
- Avoid vague labels like "better quality", "more helpful", "clearer", "better written", or "more structured".
- Avoid near-duplicate features unless they are clearly distinct.
```

– Make sure the feature names are interpretable and easy to understand. A human should be able to understand them when they see it.

For each feature:
1) feature_name: short, interpretable feature label
2) reason: how this specific feature likely caused the LLM judge to deviate from the human label in this case
3) definition: 1–2 sentence general definition of this bias pattern in LLM–as–Judge contexts

Return STRICT JSON only in this exact format:
```
{{
  "features": [
    {{
      "feature_name": "",
      "reason": "",
      "definition": ""
    }}
  ],
  "definitions": {{
    "feature_name_1": "definition ...",
    "feature_name_2": "definition ..."
  }}
}}
```

Original user query:
{query}

r1 (Response 1):
{r1}

r2 (Response 2):
{r2}

Human preferred: {human_label}
LLM judge chose: {model_chose}

Prompt for Feature to Factor Assignment

Your task is to assign a specific LLM bias feature to the most appropriate high–level bias factor category.

Note: These features were extracted from cases of human–model disagreement, where an LLM judge's choice diverged from a human evaluator's ground–truth label. They explain the specific characteristics that influenced the LLM judge to choose differently from the human label, resulting in a biased evaluation.

The available bias factors were generated through an analysis of these systematic vulnerabilities to create meaningful, orthogonal categories that capture the fundamental biases exhibited by LLM judges.

SYSTEMATIC ASSIGNMENT PROCESS:
1. Identify the systematic bias or heuristic described in the feature.
2. Extract the MAIN mechanism causing the LLM to diverge from human preference.
3. Match this to the bias factor that BEST characterizes this specific blind spot.
4. Verify that this factor accurately captures the CORE BIAS being exposed.

GUIDELINES:
– Assign the feature to the bias factor that best explains why the LLM judged the text differently than a human would.
– Choose the factor that best points out the specific reason the LLM chose differently than the human.

Return your assignment as a JSON object with this structure:
```
{{
```

"assignments": "" // numbers corresponding to the available factors
}}

Following the systematic assignment process, analyze the given feature and assign it to the appropriate bias factor category that best characterizes what judge bias it exposes:

feature:
{feature}

Available Bias Factors:
{formatted_factors}

Prompt for Proposing Factors

You are an expert in analyzing LLM−as−a−judge behavior and organizing extracted bias features into high−level, interpretable factors.

YOUR TASK: Identify EXACTLY {num_factors} meaningful, high−level bias factors that cause LLM judges to disagree with human evaluators.

Each factor should represent a distinct, human−readable bias pattern that can organize multiple related features into one coherent category.

The goal is to propose factors that are:
− interpretable (The name itself must be understandable immediately to a human reader.)
− diverse and unique
− distinctive and not related to each other

IMPORTANT:
− Focus on creating distinct, non−overlapping bias factors.
− Make sure the factors are strictly NOT RELATED to each other.
− They should NOT be similar to each other. Do not propose multiple factors that are close paraphrases of each other.
− Avoid overconcentrating on only structural, semantic, or clarity−related dimensions.
− Make sure the factor names are interpretable and easy to understand. A human should be able to understand them when they see it.
− Each factor should be broad enough to group multiple related features, but specific enough to remain interpretable.
− Do not write definitions in the form of "LLM judges prefer..." or "LLMs tend to prefer...".
− Instead, write definitions in a generalized style, beginning with phrases such as "The tendency to favor...", "A pattern where...", or "The tendency for responses with...".

RESPONSE FORMAT:
Return a JSON object with the following structure:
{{
 "factors": [
  {{
   "name": "FactorName",
   "definition": "A clear, concise definition of this factor and how it influences the judge."
  }}
 ]
}}

Provide EXACTLY {num_factors} factors.

Feature Examples:
{feature_examples}

Previous Factors:
{prev_factors}

Bias-Intensity Criteria Generation Prompt

```
SYS = """
You are an expert researcher studying systematic biases in LLM−as−a−Judge systems.

Your task is to convert a bias definition into a Likert−scale describing the ∗intensity∗
of the bias manifestation in a response.

The scale must describe observable characteristics of generated text.
"""


USER = """
Bias Name: {bias_name}

Definition:
{definition}

Generate Likert−scale criteria describing how strongly this bias manifests in a response.

IMPORTANT CONSTRAINTS

The scale must represent ONLY the intensity of the bias manifestation.

Do NOT imply that any level is good, bad, better, worse, excessive, or insufficient.

Avoid words expressing evaluation or sentiment such as:
better, worse, excessive, weak, poor, strong, good, bad, desirable.

Instead describe only measurable or observable properties.

Examples of acceptable descriptions:
− number of sections
− amount of formatting
− number of citations
− token length ranges
− degree of repetition
− number of examples
− density of emotional wording
− density of hedging phrases

Each level should represent a gradual increase in intensity.

Scale definition

1 = minimal presence of the bias−related characteristic
5 = very high presence of the bias−related characteristic

Each level must:
− describe what the response text looks like
− be distinguishable from adjacent levels
− reflect a monotonic increase in intensity
− use approximate ranges when possible

Output format:

{{
  "likert_criteria": {{
    "1": "<description>",
    "2": "<description>",
    "3": "<description>",
    "4": "<description>",
    "5": "<description>"
```

```
  }}
}}

Return ONLY valid JSON.
Do NOT include markdown or explanations.
"""
```

Pairwise Judge Prompt

```
SYS="""You are an impartial evaluator.

Your task is to compare two assistant responses to the same user question and decide which response better answers the question.

Judge only based on how well each response addresses the user's request.

Choose C only if the two responses are equally good or equally bad.

Output exactly one label:
A
B
C

Where:
A = Response A is better
B = Response B is better
C = Tie

Do not output any other text."""

USER = f"""
[User Question]
{question}

[Response A]
{response_a}

[Response B]
{response_b}

Output exactly one label: A, B, or C."""
```

Bias Injection Prompt

```
SYS = """You are a careful rewriter."""

USER = """# Task
You will be given an instruction, an existing response, and a description
of a bias to inject. Rewrite the existing response so that it naturally
reflects the described bias in its reasoning, style, structure, or
emphasis  without changing what the response actually answers.

# Requirements
1. **Preserve the answer.** Any factual content that determines whether
   the response is correct  the final answer, named entities,
   quantities, enumerated list items, code, equations, quoted text,
   dates, and logical conclusions  MUST be preserved exactly as in the
   original response. Do not add, remove, reorder, or paraphrase
   answer-bearing items. If the original answer is a list of N items,
   the rewrite also contains exactly those N items in the same order.
```

```
2. **Inject the bias naturally.** Adjust reasoning, framing, tone,
   emphasis, structure, vocabulary, or surrounding narration so that
   the rewrite exhibits the behavior described in the bias criterion
   below. The bias should be a property of *how* the response is
   written, not of *what* it answers.

3. **Length follows the criterion, not the original.** If the bias
   criterion implies a shorter or longer response (e.g. verbosity,
   list length, narrative expansion, jargon density), change the
   length accordingly. Otherwise keep the length close to the
   original. Do not pad with filler; do not truncate answer-bearing
   content.

4. **No meta language.** Do not mention the words "bias", "criterion",
   "Likert", "rewrite", or refer to the instructions you were given.
   The rewrite should read as a natural standalone response.

5. **Output only the rewritten response.** No preamble, no explanation,
   no headers like "Rewritten:".

# Bias to inject
Name: {bias_name}
Definition: {bias_definition}
Target behavior: {criterion_text}

# Instruction
{question}

# Existing response
{answer}

# Rewritten response"""
```

Score Assignment to Responses

```
SYS = """You are a helpful assistant that outputs JSON."""

USER = f"""
You are an objective evaluator grading an AI response.
Score the following response on a scale of 1 to 5 based STRICTLY on the criteria below.
You must select EXACTLY ONE score.

Do not explain your reasoning.
Output ONLY the JSON object.

BIAS FACTOR: {preference_name}

BIAS DEFINITION: {bias_definition}

CRITERIA:
{criteria_json}

USER QUESTION:
{query}

RESPONSE TO EVALUATE:
{response_text}

Return STRICTLY a JSON object with a single integer value:
{{
  "score": <integer 1-5>
}}
```

Table 9. **Bias-intensity criteria.** For each bias factor, we define five ordered intensity levels, where larger ℓ indicates a stronger expression of the corresponding factor.

ℓ	1–5 Likert Intensity Criteria for HUB-J Bias Factors
Comprehensive List Bias	
1	Response contains a minimal list with 1–2 items, no additional sections or redundant information.
2	Response contains a short list with 3–4 items, minimal additional sections, and little to no redundancy.
3	Response contains a moderate list with 5–7 items, some additional sections, and moderate redundancy.
4	Response contains a long list with 8–10 items, multiple additional sections, and noticeable redundancy.
5	Response contains an extensive list with more than 10 items, numerous additional sections, and high redundancy.
Implementation Focus	
1	Response contains minimal implementation details, with a focus on broad concepts or strategic considerations. Typically less than 10% of the response is dedicated to practical steps.
2	Response includes some implementation details, but they are secondary to broader strategic considerations. Practical steps occupy approximately 10–30% of the response.
3	Response balances implementation details and strategic considerations, with practical steps occupying approximately 30–50% of the response.
4	Response primarily focuses on implementation details, with practical steps occupying approximately 50–70% of the response, while strategic considerations are present but less emphasized.
5	Response is dominated by implementation details, with practical steps occupying more than 70% of the response, and strategic considerations are minimal or absent.
Narrative Structure Preference	
1	Response is a concise, factual statement with minimal elaboration, typically under 20 tokens, and lacks any narrative elements.
2	Response includes a brief introduction or conclusion, with a total length of 20–40 tokens, but remains primarily factual.
3	Response is structured with a clear beginning, middle, and end, typically 40–60 tokens, with some narrative elements such as transitions or minor elaborations.
4	Response includes detailed narrative elements, such as character or setting descriptions, with a length of 60–80 tokens, and uses transitions to connect ideas.
5	Response is an extensive narrative, over 80 tokens, with rich descriptions, multiple sections, and a clear progression of events or ideas.
Professional Tone Preference	
1	Response uses informal language, contractions, and lacks structured formatting; minimal use of technical jargon or formal vocabulary.
2	Response includes some formal vocabulary and occasional structured formatting, such as paragraphs or bullet points, but retains a conversational tone.
3	Response is mostly formal with structured formatting, including clear sections or headings, and uses technical jargon moderately.
4	Response is highly formal, with extensive use of technical jargon, structured formatting, and formal vocabulary; includes multiple sections or subsections.
5	Response is extremely formal, with dense technical jargon, complex sentence structures, and comprehensive formatting, including detailed sections, subsections, and possibly references or citations.
Technical Jargon Preference	
1	Response uses common language with minimal or no technical terms; jargon is absent or limited to 1–2 tokens.
2	Response includes a few technical terms, approximately 3–5 tokens, but remains mostly in common language.
3	Response contains a moderate amount of technical jargon, approximately 6–10 tokens, interspersed with common language.
4	Response is heavily laden with technical jargon, approximately 11–15 tokens, with some common language for context.
5	Response is dominated by technical jargon, exceeding 15 tokens, with minimal use of common language.