WHEN PROPERTIES ARE NOT SEPARATED DO LLMS FAIL AT JUDGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluation of Large Language Models (LLMs) is critical for reliable deployment, and the LLM-as-a-Judge paradigm is widely used for scalable evaluation involving verifiable factual properties and subjective properties. Prior work uses a single judge model or loosely coordinated multi-agent models that produce a holistic score, which empirical works claim to be unreliable under distribution shift and subject to biases such as self-preference and sensitivity to output length; recent works mitigate these issues through prompting, ensembling, or tool augmentation, but retain the same structure by combining verifiable factual properties and subjective properties within a single evaluation process. We hypothesize that evaluation error is governed by verifiable factual properties coverage α, where lower α increases reliance on subjective properties, and propose **`Judgment Decomposition Graphs (JDGs)`** to increase α by representing evaluation as a directed acyclic graph $G = (V, E)$ over evaluative properties in four phases: Phase I extracts and normalizes properties, assigns verification types $\tau(p_i) \in \{0, 1\}$, and constructs G; Phase II applies executable verifiable factual properties and subjective properties according to $\tau(p_i)$; Phase III aggregates node-level outputs into a scalar score; and Phase IV calibrates the score as a function of verifiable factual properties coverage α. Expected evaluation error is bounded by $(1 - \alpha)\,\varepsilon_{\text{LLM}} + \alpha\,(1 - \delta)$, where δ denotes executable verification reliability. Across *ReviewMT* and *JudgeBench*, **`JDGs`** achieves HA of $0.82/0.88$, reduces SPB by up to 46.6%, and lowers CE by up to 43.1% over baselines. Increasing verification coverage α improves HA from $0.69 \rightarrow 0.83$ and $0.75 \rightarrow 0.89$, while iterative optimization reduces the proxy gap from $0.20 \rightarrow 0.03$ and robustness drift from $0.124 \rightarrow 0.061$.

1 INTRODUCTION

Evaluation of Large Language Models (LLMs) is critical for reliable deployment, particularly in high-stakes domains such as scientific research Sheikhi et al. (2026), software engineering Chondamrongkul et al. (2025), and decision support Chittimalla & Potluri (2025), where incorrect or poorly evaluated outputs can lead to downstream errors Wang et al. (2026a); automated evaluation is therefore central to LLM pipelines for benchmarking model capabilities Hu et al. (2025) and guiding iterative improvement Xue et al. (2025). One widely used approach is the LLM-as-a-Judge paradigm Feng et al. (2025), where a language model acts as a scoring function $J(x, y) \rightarrow s$, assigning a score s to a candidate output y given an input x as a cost-effective alternative to human judging. In practice[1], evaluation involves both verifiable factual properties and subjective properties Ruan et al. (2026), where $\alpha \in [0, 1]$ denotes the fraction of verifiable factual properties and $(1 - \alpha)$ the fraction of subjective properties.

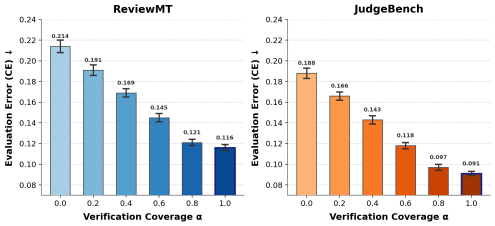

Figure 1: Empirical analysis of evaluation error as a function of verification coverage α, where error increases as α decreases, indicating greater reliance on subjective properties.

[1] Verifiable factual properties are those whose correctness can be determined through deterministic procedures or external tools, such as numerical correctness, consistency, and reproducibility, whereas subjective properties, including novelty, significance, and overall quality, require interpretive judgment and cannot be directly verified.

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

Prior work performs evaluation using a single judge model Saha et al. (2025) or loosely coordinated multi-agent models Zhang et al. (2026) that produce a holistic score $s = J(x, y)$ without separating verifiable factual properties from subjective properties, instead combining them within a single forward pass Zhang et al. (2026) or an aggregated output with no difference between parts grounded in deterministic procedures González-Potes et al. (2026), external tools, or interpretive judgment Chalmers et al. (2026). Empirical works (e.g., Hasan et al. (2026)) show that such models are unreliable under distribution shift and show biases such as self-preference and sensitivity to output length, where models favor outputs from the same family Spiliopoulou et al. (2025) or assign higher scores to longer outputs regardless of correctness You et al. (2025). Recent works attempt to mitigate these issues through prompting Yang et al. (2025), ensembling Carvalho et al. (2025), or tool augmentation Zhang et al. (2025), but retain the same evaluation structure, leaving verifiable factual properties and subjective properties combined within a single evaluation process and treating evaluation as a single score $s = J(x, y)$ with no control over the fraction α of verifiable factual properties. This prevents attribution of errors to either property type and leaves the same errors unresolved.

We hypothesize that evaluation error is governed by the fraction of verifiable factual properties, α, where lower α increases reliance on subjective properties and leads to higher error (see Figure 1); improving evaluation therefore requires separating verifiable factual properties from subjective properties, assigning verifiable factual properties to executable procedures such as arithmetic consistency checks and code execution, and restricting model-based judgment to subjective properties. To operationalize this hypothesis, we propose **`Judgment Decomposition Graphs (JDGs)`**, a four-phase framework that represents evaluation as a directed acyclic graph $G = (V, E)$ over evaluative properties, with nodes corresponding to properties and edges encoding dependencies. Phase I extracts and normalizes properties, assigns verification types $\tau(p_i) \in \{0, 1\}$, and constructs G; Phase II applies executable verification and model-based judgment according to $\tau(p_i)$; Phase III aggregates node-level outputs into a scalar score; and Phase IV calibrates the score as a function of verification coverage α. Under this decomposition, expected evaluation error is bounded by $(1 - \alpha)\,\varepsilon_{\mathrm{LLM}} + \alpha\,(1 - \delta)$, where $\varepsilon_{\mathrm{LLM}}$ denotes subjective evaluation error and δ denotes executable verification reliability. In summary, the main contribution of this work are as follows:

- We hypothesize that evaluation error is governed by verification coverage α, where lower α increases reliance on subjective model-based judgment, and propose **`JDGs`**, a four-phase framework representing evaluation as a directed acyclic graph over evaluative properties with verification types $\tau(p_i) \in \{0, 1\}$, under which expected evaluation error is bounded by $(1 - \alpha)\,\varepsilon_{\mathrm{LLM}} + \alpha\,(1 - \delta)$.
- Empirically, increasing verification coverage α from 0.0 to 1.0 improves HA from 0.69 to 0.83 on ***ReviewMT*** and from 0.75 to 0.89 on ***JudgeBench***, while reducing robustness drift from 0.141 to 0.056 and from 0.126 to 0.047, respectively.

2 Related Works

Judge Models. As discussed in §1, LLM-as-a-Judge evaluation uses models such as `GPT-4` Huang et al. (2025), `GPT-4o` Huang et al. (2025), `Claude` Cassola-Bacallao et al. (2026), `LLaMA-2` Sun et al. (2026), `LLaMA-3` Adib et al. (2025), and `Mixtral` Croxford et al. (2025) to assign scores $s = J(x, y)$ through pairwise ranking Wang et al. (2026b) or scalar scoring frameworks such as `MT-Bench` Li et al. (2026) and `Chatbot Arena` Min et al. (2025). These models evaluate (x, y) through a single scoring function without decomposing evaluation into evaluative properties $\mathcal{P}$, assigning verification types $\tau(p_i)$, or representing dependencies through a graph $G = (V, E)$; although Chain-of-Thought prompting Li et al. (2025b), self-consistency Taubenfeld et al. (2025), and model ensembling Masum et al. (2025) modify the inference procedure, the final evaluation remains a direct mapping from (x, y) to a scalar score, leaving executable verification, subjective judgment, and their relative contribution combined and verification coverage α implicit.

Judge Systems As discussed in §1, recent works study multi-agent Liang et al. (2025) and tool-augmented Ma et al. (2025) evaluation models that distribute evaluation across components $\{J_1, \ldots, J_k\}$ with roles such as critique, verification, and aggregation across models including `GPT-4` Huang et al. (2025), `Claude` Cassola-Bacallao et al. (2026), and `LLaMA-3` Adib et al. (2025). External mechanisms such as code execution Mai et al. (2025), retrieval-based verification Liu et al. (2025), and programmatic checks Schmidberger et al. (2025) provide factual grounding, but

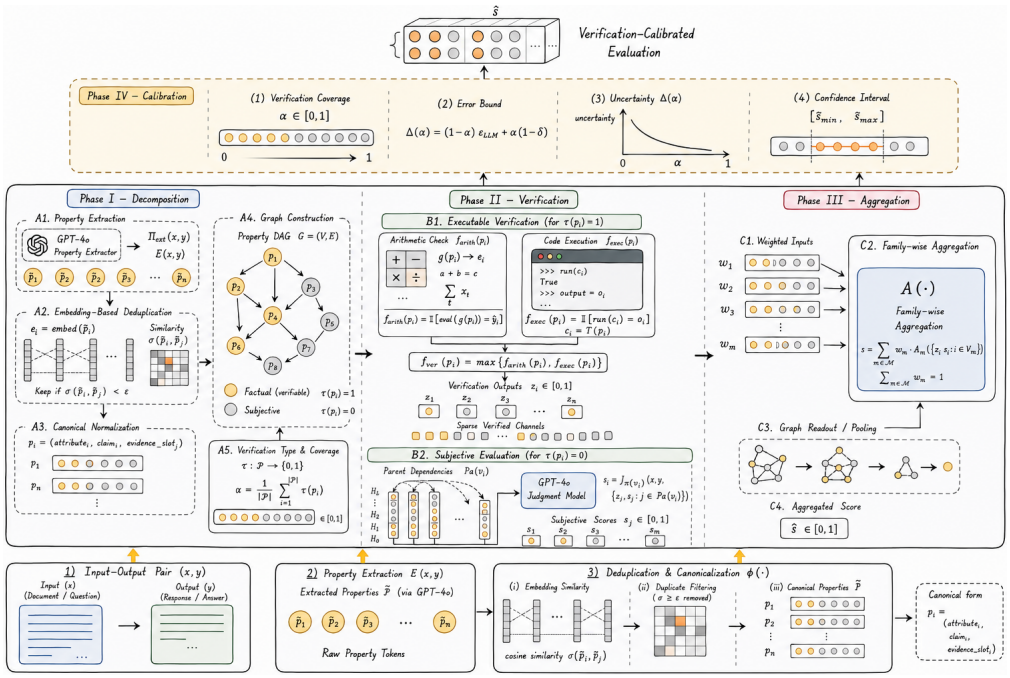

Figure 2: **`Judgment Decomposition Graphs (JDGs)`** framework with four phases: property extraction and graph construction (Phase I), executable verification and model-based judgment (Phase II), score aggregation (Phase III), and calibration using verification coverage α (Phase IV).

outputs are still aggregated into a final scalar score without constructing a property-level representation $\mathcal{P}$ with verification assignments $\tau(p_i)$ over a dependency graph $G = (V, E)$, leaving executable verifiable factual properties and subjective properties combined and the effect of verification coverage α on evaluation reliability unmodeled.

3 METHODOLOGY

We formalize evaluation over an input-output pair (x, y) using **`JDGs`** (see Figure 2) as a structured process over evaluative properties rather than a single mapping $s = J(x, y)$, where properties $\mathcal{P}$ are classified as verifiable factual or subjective and represented in a directed acyclic graph $G = (V, E)$ with nodes corresponding to properties and edges encoding dependencies. Node-level evaluations are aggregated into a scalar score, while verification coverage α quantifies the fraction of properties evaluated through executable verification in Phases I-IV with Lemma 1 defining the associated verification-coverage error bound.

Phase I (Decomposition). In Phase I, properties are derived via a property extraction function $E(\cdot)$, where J_{ext} denotes the language model instantiated using `GPT-4o`[2] and $\Pi_{\text{ext}}(x, y)$ is a structured prompt (see Figure 3). The property sets are defined in Equations (1) and (2), respectively (see §6), where the function $\phi(\cdot)$ is implemented as a two-stage process consisting of embedding-based deduplication followed by canonical normalization (see §6).

$$\tilde{\mathcal{P}} = E(x, y) = J_{\text{ext}}(\Pi_{\text{ext}}(x, y)) \tag{1}$$

$$\mathcal{P} = \phi(\tilde{\mathcal{P}}) = \{\phi(\tilde{p}_i)\}_{i=1}^{n} \tag{2}$$

Given $\tilde{\mathcal{P}}$, each property $\tilde{p}_i$ is embedded as $\mathbf{e}_i = \text{embed}(\tilde{p}_i)$ using `text-embedding-3-large`[3]. Pairwise similarity is computed as $\sigma(\tilde{p}_i, \tilde{p}_j) = \cos(\mathbf{e}_i, \mathbf{e}_j) = \frac{\mathbf{e}_i^\top \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\|}$, and near-duplicate properties are removed by retaining $\mathcal{P}' \subseteq \tilde{\mathcal{P}}$ such that $\sigma(p_i, p_j) < \epsilon$ for all $p_i, p_j \in \mathcal{P}'$, where $\epsilon \in [0, 1]$ is a

[2] `https://openrouter.ai/openai/gpt-4o`. We use a fixed model family across all runs so that variation in the resulting property set $\mathcal{P}$ arises from (x, y) rather than the underlying model distribution.

[3] `https://openrouter.ai/openai/text-embedding-3-large`. We use embedding-based similarity to capture semantic equivalence beyond lexical overlap, where cosine similarity in $\mathbb{R}^d$ approximates semantic similarity. The choice of `text-embedding-3-large` provides stable geometric representations across paraphrases under the constraint $\sigma(p_i, p_j) < \epsilon$ (see §6).

Step 1: Property Extraction via $\Pi_{\text{ext}}(x, y)$. Input-output pair: *"The paper reports 92% accuracy and claims statistically significant improvement over prior methods."* Extracted properties: $\tilde{\mathcal{P}} = \{\tilde{p}_1 : \texttt{reported accuracy is 92\%}, \tilde{p}_2 : \texttt{improvement is statistically significant}\}$
Step 2: Embedding-Based Deduplication. Properties are embedded as $\mathbf{e}_i = \text{embed}(\tilde{p}_i)$ and compared using cosine similarity $\sigma(\tilde{p}_i, \tilde{p}_j)$. Near-duplicate properties satisfying $\sigma(\tilde{p}_i, \tilde{p}_j) \geq \epsilon$ are merged into the filtered set $\mathcal{P}' \subseteq \tilde{\mathcal{P}}$.
Step 3: Canonical Normalization via $\phi(\cdot)$. Each property is mapped into the canonical representation $p_i = (\texttt{attribute}_i, \texttt{claim}_i, \texttt{evidence_slot}_i)$. Example: $p_1 = (\texttt{Results, accuracy = 92\%, Table 2})$.

Figure 3: Phase I decomposition pipeline in **JDGs**. Given (x, y), $E(x, y)$ produces $\tilde{\mathcal{P}}$, which is embedded and filtered using cosine similarity $\sigma(\tilde{p}_i, \tilde{p}_j)$ to obtain $\mathcal{P}'$, then normalized by $\phi(\cdot)$ into the canonical representation $p_i = (\texttt{attribute}_i, \texttt{claim}_i, \texttt{evidence_slot}_i)$ to form $\mathcal{P}$.

similarity threshold (see §6). The resulting $\mathcal{P}'$ is normalized to $\mathcal{P} = \{p_i\}$ using the canonical representation $p_i = \phi(p_i) = (\texttt{attribute}_i, \texttt{claim}_i, \texttt{evidence_slot}_i)$, as illustrated in Figure 3 (see §6).

This normalized set $\mathcal{P}$ defines the units for subsequent evaluation. Given $\mathcal{P}$, each property is assigned a *verification eligibility* via $\tau : \mathcal{P} \to \{0, 1\}$, where $\tau(p_i) = 1$ denotes a property admitting one of the implemented executable verification procedures and $\tau(p_i) = 0$ denotes a property evaluated through model-based judgment. The assignment $\tau(p_i)$ is a deterministic rule-based mapping over the canonical property representation, where $\tau(p_i) = 1$ if its attribute, claim, and evidence slot admit predefined executable verification procedures $f_{\text{arith}}(\cdot)$ or $f_{\text{exec}}(\cdot)$; this includes properties involving arithmetic correctness, numerical consistency, or executable code, while properties that do not admit these procedures receive $\tau(p_i) = 0$. To distinguish verification eligibility from realized verification, we define $r_i \in \{0, 1\}$ as an execution indicator, where $r_i = 1$ denotes that property p_i is routed through an executable verification procedure and $r_i = 0$ denotes that it remains under model-based judgment, with $r_i \leq \tau(p_i)$. We therefore define realized verification coverage as $\alpha = \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} r_i$, where $\tau(p_i)$ characterizes whether a property admits executable verification and α measures the fraction for which executable verification is actually performed. For each verifiable property, at most one executable verification procedure is designated as applicable, so $f_{\text{arith}}(\cdot)$ and $f_{\text{exec}}(\cdot)$ constitute mutually exclusive verification pathways, with combined verification defined as $f_{\text{ver}}(p_i) = \max\{f_{\text{arith}}(p_i), f_{\text{exec}}(p_i)\}$ (see §6). Evaluation is represented as a directed acyclic graph $G = (V, E)$ with $v_i \leftrightarrow p_i$, where candidate edges $(v_i, v_j) \in E$ are constructed using a directed dependency function $D(p_i, p_j)$ (see §6). Since semantic similarity alone does not determine dependency direction, $D(p_i, p_j)$ is implemented as a directed dependency classifier that predicts whether evaluation or interpretation of p_j requires information from p_i, with an edge retained when $D(p_i, p_j) \geq \gamma$. Candidate dependency edges are subsequently filtered to remove cycles, yielding an acyclic graph that admits a valid evaluation order for Phase II. This completes Phase I by constructing the property set $\mathcal{P}$, graph structure G, verification eligibility τ, and execution indicators r.

Phase II (Verification). In Phase II, nodes with $r_i = 1$ are evaluated using executable verification functions over the canonical representation from Phase I, producing $z_i = f_{\text{ver}}(p_i)$, where $f_{\text{ver}}(p_i) = \max\{f_{\text{arith}}(p_i), f_{\text{exec}}(p_i)\}$. Arithmetic verification is defined as $f_{\text{arith}}(p_i) = \mathbb{I}\big[\text{eval}\big(g(p_i)\big) = \hat{y}_i\big]$, where $g : \mathcal{P} \to \mathcal{E}$ maps p_i to a symbolic expression e_i derived from $(\texttt{attribute}_i, \texttt{claim}_i, \texttt{evidence_slot}_i)$ and $\hat{y}_i$ is the referenced value extracted from p_i; the expression is evaluated deterministically and compared with $\hat{y}_i$ to produce the binary verification result. Code-based verification is defined as $f_{\text{exec}}(p_i) = \mathbb{I}[\text{run}(c_i) = o_i]$, where $c_i = T(p_i)$ is executable code generated via $T : \mathcal{P} \to \mathcal{C}$ and o_i is the expected output extracted from p_i; for executable properties, $T(\cdot)$ translates the canonical representation into Python code, which is executed in a sandbox with a 5-second timeout and compared with o_i. Nodes with $r_i = 0$ are evaluated as $s_i = J_{\pi(v_i)}\big(x, y, \{z_j, s_j : v_j \in \text{Pa}(v_i)\}\big)$, where $\pi : V \to \mathcal{M}$ assigns nodes to the available judge model family $\mathcal{M} = \{\texttt{GPT-4o}\}$. The outputs $\{z_i\}_{r_i=1}$ and $\{s_i\}_{r_i=0}$ define node-level evaluations over $G = (V, E)$ and are aggregated in Phase III.

Phase III (Aggregation). In Phase III, node-level outputs from Phase II are aggregated into a scalar evaluation score. Let $\mathcal{R} = \{i : r_i = 1\}$ and $\mathcal{J} = \{i : r_i = 0\}$ denote the indices of properties evaluated by executable verification and model-based judgment, respectively. The dependency graph does not directly alter the aggregation weights but conditions node-level subjective judgments on the outputs of their parent properties, so aggregation over $G = (V, E)$ operates on graph-conditioned node scores and is defined as $s = A\big(\{z_i\}_{i \in \mathcal{R}}, \{s_i\}_{i \in \mathcal{J}}\big)$, where each verified output $z_i \in \{0, 1\}$ is used directly and each model-based judgment s_i is normalized to $[0, 1]$. Since the current implementation uses a single judge model family $\mathcal{M} = \{\texttt{GPT-4o}\}$, family-specific weighting is degenerate and is therefore not introduced. The final aggregated score is defined as $\hat{s} = \frac{1}{|\mathcal{P}|}\left(\sum_{i \in \mathcal{R}} z_i + \sum_{i \in \mathcal{J}} s_i\right)$,

while $\alpha = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|} r_i$ determines the fraction of properties evaluated through the executable pathway. Thus, the aggregation procedure remains fixed across verification-coverage conditions, and α affects the evaluation pathway of individual properties without introducing a separate model-family weighting term.

Phase IV (Uncertainty Calibration). In Phase IV, the aggregated score $\hat{s}$ is not modified; instead, the procedure quantifies uncertainty around the point estimate as a function of realized verification coverage α using $[\hat{s}_{\min}, \hat{s}_{\max}] = \hat{s} \pm \Delta(\alpha)$, where $\Delta(\alpha) = (1-\alpha)\,\varepsilon_{\text{LLM}} + \alpha\,(1-\bar{\delta})$, ε_{LLM} denotes the estimated error of model-based judgments, and $\bar{\delta}$ denotes the average reliability of executable verification over the properties actually routed through verification. Given $\hat{s}$ and α, calibration computes $\Delta(\alpha)$ and constructs $\hat{s}_{\min} = \hat{s} - \Delta(\alpha)$ and $\hat{s}_{\max} = \hat{s} + \Delta(\alpha)$. In the controlled verification-coverage experiment, α is varied over $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ by routing progressively larger subsets of properties satisfying $\tau(p_i) = 1$ through the verification pathway, while (x, y), $\mathcal{P}$, τ, G, and the aggregation procedure remain fixed. Since verification reliability can vary across properties, δ_i denotes the reliability of executable verification for property p_i among the properties actually routed through verification, and $\bar{\delta}$ is the average reliability over this verified subset. Thus, calibration is applied after aggregation and uses realized verification coverage to quantify uncertainty around the final point estimate rather than producing a new point estimate. Lemma 1 provides the conditional bound underlying this uncertainty radius; when $\varepsilon_{\text{LLM}} > 1 - \bar{\delta}$, the bound decreases monotonically with α.

Lemma 1 (Verification Coverage Error Bound). Let $\mathcal{R} = \{i : r_i = 1\}$ and $\mathcal{J} = \{i : r_i = 0\}$ denote the properties evaluated by executable verification and model-based judgment, respectively. Assume $\mathbb{E}[e_i \mid r_i = 0] \le \varepsilon_{\text{LLM}}$ and, for each $i \in \mathcal{R}$, $\mathbb{E}[e_i \mid r_i = 1] \le 1 - \delta_i$, where $\bar{\delta} = \frac{1}{|\mathcal{R}|}\sum_{i\in\mathcal{R}} \delta_i$ is the average reliability of executable verification over the verified subset. Then the expected mean property-level evaluation error satisfies $\text{error} \le (1-\alpha)\,\varepsilon_{\text{LLM}} + \alpha\,(1-\bar{\delta})$, where $\alpha = \frac{|\mathcal{R}|}{|\mathcal{P}|}$.

Proof. Let e_i denote the error contribution associated with property p_i. Since properties are partitioned into model-based judgment $\mathcal{J} = \{i : r_i = 0\}$ and executable verification $\mathcal{R} = \{i : r_i = 1\}$, the total mean property-level evaluation error is $\text{error} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{J}} e_i + \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{R}} e_i$, where $\alpha = \frac{|\mathcal{R}|}{|\mathcal{P}|}$ gives $|\mathcal{R}| = \alpha|\mathcal{P}|$ and $|\mathcal{J}| = (1-\alpha)|\mathcal{P}|$. Assuming $\mathbb{E}[e_i \mid r_i = 0] \le \varepsilon_{\text{LLM}}$ for model-based judgment and $\mathbb{E}[e_i \mid r_i = 1] \le 1 - \delta_i$ for executable verification, where $\bar{\delta}$ is the average of the property-specific reliabilities over $\mathcal{R}$, the partitioned error is bounded by $\text{error} \le (1-\alpha)\,\varepsilon_{\text{LLM}} + \alpha\,(1-\bar{\delta})$. Thus, as α increases, a larger fraction of evaluation is assigned to executable verification rather than model-based judgment. The bound is a property-level expected-error bound and does not, without an additional error-preservation assumption on the aggregation operator, directly imply an upper bound on the final scalar calibration error CE. Accordingly, CE is treated as an empirical evaluation metric for the final point estimate, while $\Delta(\alpha)$ characterizes the uncertainty associated with the aggregated score. □

4 Experimental Setup

Dataset. We evaluate **JDGs** on *ReviewMT* and *JudgeBench*, which contain mixtures of verifiable factual and subjective properties. **(i) ReviewMT**[4] Tan et al. (2024) contains 26,841 papers and 92,017 reviews spanning ICLR, Nature Communications, NeurIPS, ICML, CVPR, ICCV, and ACL, with reviewer reports, rebuttal discussions, review trajectories, scores, confidence scores, and final decisions. It includes verifiable factual properties such as numerical consistency, methodological correctness, citation-grounded statements, reproducibility, executable claims, and score updates, and subjective properties such as novelty, significance, clarity, contribution quality, reviewer confidence, and overall assessment. **(ii) JudgeBench**[5] Tan et al. (2025) contains 350 response pairs across mathematical reasoning, code reasoning, tool use, and knowledge-intensive tasks, with objective correctness labels, including verifiable factual properties such as arithmetic correctness, execution correctness, symbolic consistency, and tool-output agreement, and subjective properties such as reasoning quality, coherence, and explanation completeness.

Verification Assignment Validation. **JDGs**' deterministic rule-based verification assignment $\tau(p_i)$ is evaluated against human judgments to assess the consistency of the distinction between verifiable factual and subjective properties, with human annotations used only for validation and not to generate or modify $\tau(p_i)$. For each dataset, we randomly sample 500 properties and obtain binary labels from 3 independent annotators indicating whether each property is verifiable factual or subjective, and report Precision, Recall, F1, and Accuracy against the majority-vote human labels.

Evaluation Metrics. Following prior works (e.g., Li et al. (2025a)) on LLM-as-a-Judge evaluation, calibration, and judge bias, we evaluate **JDGs** using four metrics. *(i) Human Agreement (HA)* is measured using Spearman rank correlation $\rho = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2-1)}$, where $d_i = \text{rank}(\hat{s}_i) - \text{rank}(s_i)$, with higher ρ indicating stronger agreement with human judgments; for *ReviewMT*, s_i is the normalized

[4] https://github.com/chengtan9907/ReviewMT

[5] https://github.com/ScalerLab/JudgeBench

Table 1: Comparison of **JDGs** against backbone judge models and prior judge frameworks on *ReviewMT* and *JudgeBench*.

Method	ReviewMT					JudgeBench				
	HA ↑	SPB ↓	CE ↓	Ro ↓	VC (%) ↑	HA ↑	SPB ↓	CE ↓	Ro ↓	VC (%) ↑
`GPT-4o`	0.71	18.4	0.193	0.124	34.8	0.78	15.9	0.171	0.108	41.2
`Claude-3.5-Sonnet`	0.69	16.7	0.201	0.118	33.5	0.76	14.8	0.176	0.101	39.6
`LLaMA-3.1-8B-Instruct`	0.61	24.9	0.248	0.183	27.4	0.68	22.3	0.221	0.165	31.7
`MT-Bench` Bai et al. (2024)	0.65	21.1	0.229	0.169	29.8	0.71	19.5	0.207	0.148	34.1
`Chatbot Arena` Min et al. (2025)	0.67	19.4	0.214	0.157	31.2	0.73	17.8	0.193	0.139	35.6
`JudgeLM` Zhu et al. (2025)	0.70	17.9	0.198	0.132	36.9	0.77	15.4	0.179	0.116	42.7
`PandaLM` Wang et al. (2024)	0.68	18.6	0.206	0.141	35.1	0.75	16.2	0.184	0.124	40.8
JDGs (Ours)	**0.82**	**9.7**	**0.121**	**0.061**	**78.4**	**0.88**	**7.9**	**0.097**	**0.052**	**84.6**

reviewer score associated with each evaluated review, while for *JudgeBench*, s_i corresponds to human-annotated correctness labels. *(ii) Self-Preference Bias (SPB)* is defined as $\mathrm{SPB} = S_{\mathrm{same}} - S_{\mathrm{other}}$, where S_{same} denotes the mean judge score assigned to outputs generated by the judge's own model family and S_{other} denotes the corresponding mean score for outputs from other model families, with the comparison performed on matched outputs with equivalent task correctness and quality. *(iii) Calibration Error (CE)* is $\mathrm{CE} = \frac{1}{N}\sum_{i=1}^{N} |\hat{s}_i - s_i|$, measuring alignment between calibrated **JDGs** scores and human reference labels. *(iv) Robustness (Ro)* under iterative evaluation is measured as $\Delta_{\mathrm{drift}} = \frac{1}{T}\sum_{t=1}^{T} |\hat{s}^{(t)} - \hat{s}^{(0)}|$, with lower values indicating greater resistance to Goodhart-style drift[6]. Verification Coverage (VC) is reported as $\alpha = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|} r_i$, where $r_i \in \{0, 1\}$ indicates whether property p_i is actually routed through executable verification.

Hyperparameters. All experiments use `GPT-4o` for property extraction $E(\cdot)$ and subjective node evaluation $J_{\pi(v_i)}(\cdot)$, and `text-embedding-3-large` for Phase I deduplication, with $\epsilon = 0.85$, $\gamma = 0.7$, and $\eta = 5$s for embedding similarity, dependency edges, and arithmetic/sandboxed executable verification, respectively. The Tool-Augmented Judge uses the same `GPT-4o` backbone, verification functions, and timeout without JDGs decomposition or graph components. Verification coverage uses $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ with fixed input-output pairs, properties, dependency graph, and aggregation; subjective judgments are normalized to $[0, 1]$, verified outputs to $\{0, 1\}$, and family weights to $w_m = 1/|\mathcal{M}|$. $\varepsilon_{\mathrm{LLM}}$ is estimated from subjective-judgment errors, δ_i from executable-verification outcomes, and $\bar{\delta}$ as their mean over the verifiable subset, held fixed across coverage settings. Calibration uses $\Delta(\alpha)$ without additional learned parameters, and iterative robustness evaluation uses $T = 10$ optimization steps. Experiments use $4\times$ NVIDIA A100 GPUs (80GB each), AMD EPYC CPUs, 512GB RAM, and batch size $B = 8$; sensitivity analyses for ϵ, γ, η, T, and B are reported in §6.

Baselines. We compare **JDGs** against representative LLM-as-a-Judge baselines and prior judge frameworks. Backbone judge baselines include `GPT-4o`, `Claude-3.5-Sonnet`[7], and `LLaMA-3.1-8B-Instruct`[8], each evaluated as direct scoring functions $s = J(x, y)$ under zero-shot prompting, along with `MT-Bench` Bai et al. (2024), `Chatbot Arena` Min et al. (2025), `JudgeLM` Zhu et al. (2025), and `PandaLM` Wang et al. (2024). We additionally include a Tool-Augmented Judge using `GPT-4o` for holistic scoring with the same arithmetic and executable verification functions $f_{\mathrm{arith}}(\cdot)$ and $f_{\mathrm{exec}}(\cdot)$, but without property decomposition, dependency graph construction, verification assignment $\tau(p_i)$, or verification-coverage calibration.

5 Result and Analysis

Comparison With State-of-the-Arts. Table 1 compares **JDGs** against backbone judge models and prior judge frameworks on *ReviewMT* and *JudgeBench*. Across both datasets, **JDGs** achieves the highest Human Agreement (HA) and Verification Coverage (VC), and the lowest Self-Preference Bias (SPB), Calibration Error (CE), and iterative Robustness drift (Ro). On *ReviewMT*, HA increases from 0.71 to 0.82, while SPB decreases from 16.7-18.4 to 9.7 and Ro from 0.118-0.124 to 0.061; on *JudgeBench*, **JDGs** achieves HA of 0.88 and VC of 84.6%, with VC gains of up to +112.5% and Ro reductions of up to −48.5% over non-**JDGs** baselines. The larger gap on *ReviewMT* reflects the

[6] Goodhart-style drift refers to score inflation under iterative optimization, where maximizing $J(x, y^{(t)})$ increases $\hat{s}^{(t)}$ without corresponding improvement in target quality $q(y^{(t)})$.

[7] `https://openrouter.ai/anthropic/claude-3.5-sonnet`

[8] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`

Table 5: Effect of verification coverage $\alpha = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\tau(p_i)$ on evaluation reliability. Results are averaged over 5 runs; $\pm$ denotes standard deviation.

Verification Coverage α	ReviewMT				JudgeBench			
	HA ↑	SPB ↓	CE ↓	Ro ↓	HA ↑	SPB ↓	CE ↓	Ro ↓
0.0	0.69 ± 0.01	19.6 ± 0.5	0.214 ± 0.006	0.141 ± 0.005	0.75 ± 0.01	17.8 ± 0.4	0.188 ± 0.005	0.126 ± 0.004
0.2	0.72 ± 0.01	17.2 ± 0.4	0.191 ± 0.005	0.119 ± 0.004	0.78 ± 0.01	15.3 ± 0.4	0.166 ± 0.004	0.104 ± 0.003
0.4	0.75 ± 0.01	14.8 ± 0.4	0.169 ± 0.004	0.098 ± 0.003	0.81 ± 0.01	13.1 ± 0.3	0.143 ± 0.004	0.086 ± 0.003
0.6	0.79 ± 0.01	12.3 ± 0.3	0.145 ± 0.004	0.079 ± 0.003	0.85 ± 0.01	10.6 ± 0.3	0.118 ± 0.003	0.069 ± 0.002
0.8	0.82 ± 0.01	9.7 ± 0.3	0.121 ± 0.003	0.061 ± 0.002	0.88 ± 0.01	7.9 ± 0.2	0.097 ± 0.003	0.052 ± 0.002
1.0	$\mathbf{0.83 \pm 0.01}$	$\mathbf{8.9 \pm 0.2}$	$\mathbf{0.116 \pm 0.003}$	$\mathbf{0.056 \pm 0.002}$	$\mathbf{0.89 \pm 0.01}$	$\mathbf{7.1 \pm 0.2}$	$\mathbf{0.091 \pm 0.002}$	$\mathbf{0.047 \pm 0.002}$

long-form peer-review discussions and mixed subjective assessments, which increase reliance on holistic scoring functions $s = J(x, y)$ and result in higher calibration error, stronger family-specific preference bias, and larger iterative drift. In contrast, **JDGs** decomposes evaluation into executable and subjective components through arithmetic verification $f_{\text{arith}}(\cdot)$ and executable verification $f_{\text{exec}}(\cdot)$, increasing VC from approximately 30%-40% in prior baselines to more than 78% across both datasets and reducing reliance on subjective model-based judgment and iterative Goodhart-style drift.

Human Validation of Verification Assignment. To validate the deterministic rule-based verification assignment $\tau(p_i)$, we randomly sample 500 properties from each dataset and obtain binary labels from 3 independent annotators indicating whether each property is verifiable factual or subjective. Human annotations are used only for independent validation and do not generate or modify $\tau(p_i)$. We compare majority-vote labels with **JDGs** assignments using Precision, Recall, F1, and Accuracy. As shown in Table 2, **JDGs** achieves Precision, Recall, F1, and Accuracy of 94.8%, 93.5%, 94.1%, and 94.6% on *ReviewMT*, and 95.9%, 94.6%, 95.2%, and 95.5% on *JudgeBench*, respectively, indicating high agreement with human annotations.

Table 2: Human evaluation of verification assignment on *ReviewMT* and *JudgeBench*. Prec. denotes Precision.

Dataset	Human Evaluation		Verification Assignment			
	Properties	Annotators	Prec. ↑	Recall ↑	F1 ↑	Accuracy ↑
ReviewMT	500	3	94.8%	93.5%	94.1%	94.6%
JudgeBench	500	3	95.9%	94.6%	95.2%	95.5%

Comparison With Tool-Augmented Baseline. To isolate the contribution of executable verification, we compare **JDGs** with a Tool-Augmented Judge using the same `GPT-4o` backbone and executable verification functions but omitting property decomposition, dependency graph construction, verification assignment $\tau(p_i)$, and verification-coverage calibration. As shown in Table 3, executable verification improves HA from 0.71 to 0.76 and reduces CE from 0.193 to 0.158 on *ReviewMT*, and improves HA from 0.78 to 0.83 and reduces CE from 0.171 to 0.134 on *JudgeBench*, while **JDGs** further improves HA to 0.82 and 0.88 and reduces CE to 0.121 and 0.097, respectively.

Table 3: Comparison with a tool-augmented baseline on *ReviewMT* and *JudgeBench*.

Method	ReviewMT		JudgeBench	
	HA ↑	CE ↓	HA ↑	CE ↓
`GPT-4o`	0.71	0.193	0.78	0.171
`Tool-Augmented Judge`	0.76	0.158	0.83	0.134
JDGs (Ours)	**0.82**	**0.121**	**0.88**	**0.097**

Verification Coverage Analysis. To control verification coverage α, each sample is processed once to obtain a fixed property set $\mathcal{P}$ and dependency graph G (see Table 5). We construct settings over $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ by routing progressively larger subsets of executable properties through verification while keeping the input-output pairs, properties, graph, and aggregation fixed; unrouted properties remain under the same subjective judgment procedure. Across both *ReviewMT* and *JudgeBench*, increasing α monotonically improves HA and reduces SPB, CE, and Ro: on *ReviewMT*, HA increases from 0.69 to 0.83 and CE decreases from 0.214 to 0.116, while on *JudgeBench*, HA increases from 0.75 to 0.89 and CE decreases from 0.188 to 0.091. The largest gains occur between $\alpha = 0.2$ and $\alpha = 0.8$. To examine Lemma 1, we compare $\Delta(\alpha)$ with observed CE; Table 4 shows decreasing predicted and observed errors, supporting the conditional relationship between verification coverage and evaluation error.

Table 4: Predicted and observed evaluation error across verification coverage settings on *ReviewMT*.

Verification Coverage	Evaluation Error	
α	Predicted Error ↓	Observed CE ↓
0.0	0.214	0.214
0.2	0.195	0.191
0.4	0.173	0.169
0.6	0.149	0.145
0.8	0.126	0.121
1.0	0.111	0.116

Cross-Properties Interference Analysis. Table 6 evaluates cross-property interference by calibrating models on one property type and evaluating on another. Across $\tau(p_i) = 1 \rightarrow \tau(p_i) = 0$ and

Table 6: Cross-property interference across evaluation distributions, where factual properties correspond to $\tau(p_i) = 1$ and subjective properties to $\tau(p_i) = 0$.

Train	Test	Method	ReviewMT					JudgeBench				
			HA ↑	SPB ↓	CE ↓	Ro ↓	VC (%) ↑	HA ↑	SPB ↓	CE ↓	Ro ↓	VC (%) ↑
$\tau(p_i) = 1$	$\tau(p_i) = 0$	GPT-4o	0.66	20.4	0.218	0.146	31.2	0.73	18.1	0.194	0.129	36.5
		Claude-3.5-Sonnet	0.64	19.2	0.224	0.139	30.1	0.71	17.3	0.198	0.121	35.2
		LLaMA-3.1-8B-Instruct	0.57	27.1	0.263	0.197	25.3	0.64	24.2	0.236	0.178	29.4
		MT-Bench Bai et al. (2024)	0.61	22.8	0.241	0.181	27.8	0.68	20.6	0.217	0.159	32.1
		Chatbot Arena Min et al. (2025)	0.63	21.2	0.226	0.167	28.9	0.70	18.9	0.203	0.147	33.4
		JudgeLM Zhu et al. (2025)	0.65	19.6	0.211	0.151	35.8	0.72	17.1	0.189	0.132	40.3
		PandaLM Wang et al. (2024)	0.63	20.3	0.219	0.158	33.6	0.71	17.8	0.194	0.139	38.2
		JDGs (Ours)	**0.80**	**10.3**	**0.127**	**0.066**	**74.1**	**0.87**	**8.5**	**0.102**	**0.057**	**81.7**
$\tau(p_i) = 0$	$\tau(p_i) = 1$	GPT-4o	0.64	21.7	0.229	0.158	28.4	0.71	19.4	0.207	0.141	33.1
		Claude-3.5-Sonnet	0.62	20.3	0.236	0.149	27.6	0.69	18.2	0.211	0.132	31.9
		LLaMA-3.1-8B-Instruct	0.55	28.6	0.271	0.204	22.7	0.62	25.4	0.244	0.186	27.1
		MT-Bench Bai et al. (2024)	0.59	24.1	0.247	0.188	25.8	0.66	21.7	0.223	0.167	29.6
		Chatbot Arena Min et al. (2025)	0.61	22.6	0.233	0.176	26.9	0.68	20.1	0.209	0.154	31.2
		JudgeLM Zhu et al. (2025)	0.63	20.9	0.219	0.161	32.4	0.70	18.3	0.197	0.142	37.8
		PandaLM Wang et al. (2024)	0.61	21.5	0.227	0.169	30.8	0.69	19.1	0.202	0.148	35.6
		JDGs (Ours)	**0.79**	**11.1**	**0.134**	**0.072**	**71.3**	**0.85**	**9.2**	**0.109**	**0.061**	**79.4**

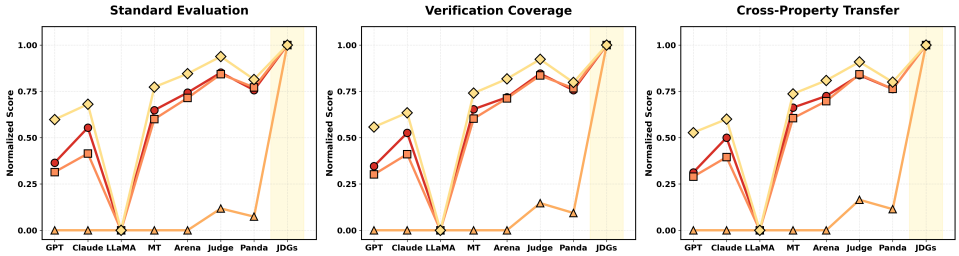

Figure 4: Computational analysis of **JDGs** and baselines across standard evaluation, verification-coverage control, and cross-property transfer.

$\tau(p_i) = 0 \rightarrow \tau(p_i) = 1$, **JDGs** achieves higher HA and VC and lower SPB, CE, and Ro than all baselines on both *ReviewMT* and *JudgeBench*. Baselines such as `GPT-4o`, `Claude-3.5-Sonnet`, `JudgeLM`, and `PandaLM`, using a single scoring function $s = J(x, y)$ without verification assignments $\tau(p_i)$, show higher calibration error and iterative drift under transfer, whereas **JDGs** maintains HA of 0.79-0.88 and VC above 71%, supporting the hypothesis that separating evaluative properties through $\tau(p_i)$ reduces cross-property interference.

Computational Analysis. Figure 4 reports the computational overhead of **JDGs** across standard evaluation, verification-coverage control, and cross-property transfer. Compared with baselines, **JDGs** incurs higher latency, memory usage, evaluation cost, and executable verification calls due to property decomposition, graph construction, and verification through $f_{\text{arith}}(\cdot)$ and $f_{\text{exec}}(\cdot)$, while overhead remains stable across verification-coverage and cross-property transfer settings. These gains in HA, CE, Ro, and VC require no increase in the underlying judge model's parameter count, although **JDGs** introduces additional inference and verification computation.

6 Ablation Study

Table 7 evaluates the contribution of each **JDGs** component to evaluation reliability. Removing property extraction $E(x, y)$ or verification assignment $\tau(p_i)$ produces the largest degradation; on *ReviewMT*, removing $\tau(p_i)$ decreases HA from 0.82 to 0.71 and increases CE from 0.121 to 0.203 and Ro from 0.061 to 0.131. Removing $f_{\text{arith}}(\cdot)$ or $f_{\text{exec}}(\cdot)$ reduces VC from 78.4% to 58.7% and 61.2%, respectively, with corresponding increases in SPB, CE, and Ro, while removing $G = (V, E)$, $A(\cdot)$, $A_m(\cdot)$, $\Delta(\alpha)$, or the Lemma 1 error bound also reduces reliability. Removing $A(\cdot)$, $A_m(\cdot)$, or $\Delta(\alpha)$ increases CE on *ReviewMT* to 0.151, 0.146, and 0.143, respectively. Full **JDGs** achieves HA of 0.82/0.88 and CE of 0.121/0.097 on *ReviewMT*/*JudgeBench*. To isolate dependency modeling, Table 8 compares **JDGs** with a property-level checklist using the same decomposition and verification assignment but without dependency edges. The checklist achieves HA of 0.79/0.85 and CE of 0.147/0.121, while **JDGs** achieves 0.82/0.88 and 0.121/0.097, indicating additional gains from dependency modeling.

Table 8: Effect of dependency graph modeling.

Method	ReviewMT		JudgeBench	
	HA ↑	CE ↓	HA ↑	CE ↓
Property Checklist + Verification	0.79	0.147	0.85	0.121
JDGs (Property Graph + Verification)	**0.82**	**0.121**	**0.88**	**0.097**

Table 7: Ablation analysis of **JDGs** on *ReviewMT* and *JudgeBench*.

Variant	ReviewMT					JudgeBench				
	HA ↑	SPB ↓	CE ↓	Ro ↓	VC (%) ↑	HA ↑	SPB ↓	CE ↓	Ro ↓	VC (%) ↑
w/o Property Extraction $E(x, y)$	0.72	18.6	0.191	0.126	36.4	0.79	16.1	0.169	0.109	41.8
w/o Embedding Deduplication $\sigma(p_i, p_j)$	0.77	13.8	0.158	0.087	65.1	0.83	11.9	0.131	0.073	71.2
w/o Canonical Normalization $\phi(\cdot)$	0.75	15.1	0.168	0.094	60.7	0.82	12.8	0.142	0.081	66.4
w/o Dependency Graph $G = (V, E)$	0.78	12.9	0.149	0.081	73.4	0.84	10.7	0.124	0.068	78.6
w/o Arithmetic Verification f_{arith}	0.75	15.2	0.171	0.098	58.7	0.82	13.1	0.146	0.083	64.5
w/o Executable Verification f_{exec}	0.76	14.6	0.166	0.094	61.2	0.83	12.4	0.139	0.079	67.1
w/o Verification Assignment $\tau(p_i)$	0.71	19.1	0.203	0.131	34.9	0.77	16.8	0.181	0.114	40.6
w/o Aggregation $A(\cdot)$	0.78	12.6	0.151	0.083	78.4	0.84	10.4	0.126	0.069	84.6
w/o Family-Aware Aggregation $A_m(\cdot)$	0.79	12.1	0.146	0.078	78.4	0.85	9.9	0.119	0.065	84.6
w/o Calibration $\Delta(\alpha)$	0.79	11.8	0.143	0.076	78.4	0.85	9.8	0.117	0.064	84.6
w/o Lemma 1 Error Bound	0.80	11.4	0.138	0.071	78.4	0.86	9.4	0.113	0.059	84.6
JDGs (Full)	**0.82**	**9.7**	**0.121**	**0.061**	**78.4**	**0.88**	**7.9**	**0.097**	**0.052**	**84.6**

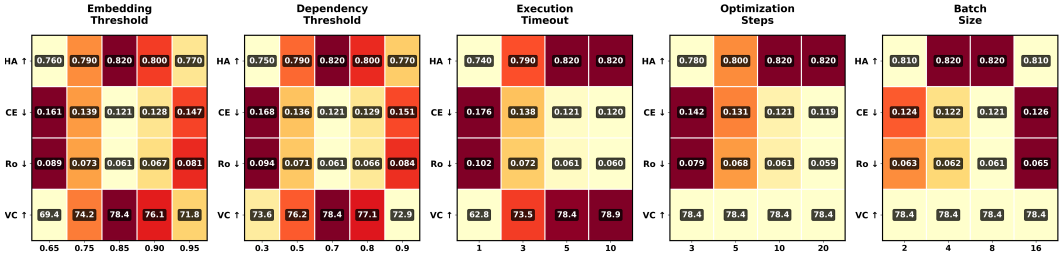

Figure 5: Hyperparameter sensitivity analysis of **JDGs** on *ReviewMT*, with each hyperparameter varied independently while others remain fixed to default values.

Hyperparameter Sensitivity Analysis. Figure 5 evaluates the sensitivity of **JDGs** to key hyperparameters in Phases I-IV. Performance remains stable across moderate variations of ϵ, γ, η, T, and B, with the best performance at $\epsilon = 0.85$, $\gamma = 0.7$, $\eta = 5$s, and $T = 10$, achieving HA of 0.82, CE of 0.121, and Ro of 0.061. Lower ϵ and γ increase redundant or weakly related properties in $G = (V, E)$, reducing HA and increasing CE, while $\eta = 1$s reduces VC from 78.4% to 62.8%. Batch-size variation produces only minor changes, indicating that calibration and robustness gains primarily arise from verification assignment $\tau(p_i)$ and executable verification rather than optimization scale alone.

Goodhart-Style Drift Analysis. Table 9 evaluates robustness under iterative optimization for $T = 10$ steps, where the candidate response $y^{(t)}$ is updated using a fixed response-optimization procedure $\mathcal{O}$ with x, $y^{(t)}$, and judge score $\hat{s}^{(t)}$ as the optimization objective, $y^{(t+1)} = \mathcal{O}\left(x, y^{(t)}, \hat{s}^{(t)}\right)$ for $t = 0, \ldots, T-1$. We record $\hat{s}^{(t)}$ and independent human quality score $q(y^{(t)})$ at each step. On *ReviewMT*, `GPT-4o` achieves $\Delta\hat{s} = 0.31$ and $\Delta q = 0.11$, yielding a proxy gap of 0.20, whereas **JDGs** reduces the gap to 0.03 with $\Delta\hat{s} = 0.17$ and $\Delta q = 0.14$, while achieving the lowest drift across both datasets.

Table 9: Goodhart-style drift analysis under iterative optimization for $T = 10$ steps.

Method	ReviewMT			
	$\Delta\hat{s}$ ↑	Δq ↑	Proxy Gap ↓	Δ_{drift} ↓
`GPT-4o`	0.31	0.11	0.20	0.124
`Claude-3.5-Sonnet`	0.29	0.10	0.19	0.118
`LLaMA-3.1-8B-Instruct`	0.37	0.08	0.29	0.183
`MT-Bench` Bai et al. (2024)	0.35	0.09	0.26	0.169
`Chatbot Arena` Min et al. (2025)	0.33	0.10	0.23	0.157
`JudgeLM` Zhu et al. (2025)	0.30	0.11	0.19	0.132
`PandaLM` Wang et al. (2024)	0.32	0.10	0.22	0.141
JDGs (Ours)	**0.17**	**0.14**	**0.03**	**0.061**

7 Conclusion

This work proposed **Judgment Decomposition Graphs (JDGs)**, a four-phase framework that decomposes evaluation into executable and subjective properties through verification assignments $\tau(p_i)$. Across *ReviewMT* and *JudgeBench*, **JDGs** improved human agreement, reduced calibration error, self-preference bias, and Goodhart-style drift relative to baselines. Verification coverage control, cross-property transfer, and ablation studies support the hypothesis that increasing verification coverage α reduces evaluation error. Limitations include additional computational overhead and evaluation restricted to English-language settings and fixed verification functions; future work will study multilingual evaluation and broader verification mechanisms.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish the writing of the manuscript. We have not used generative AI tools for generating or modifying experimental results, or scientific claims, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work and verified the final content. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve collecting new personal or sensitive data. Human annotations were used to validate the verification assignment $\tau(p_i)$ and obtain human quality scores $q(\cdot)$ for Goodhart-style drift analysis. Three NLP researchers (aged 25-30) with prior experience in LLM evaluation and peer-review analysis independently annotated sampled instances from *ReviewMT* and *JudgeBench*-without compensation by male members of the same research group, with disagreements resolved through majority agreement; Cohen's κ was 0.81 on *ReviewMT* and 0.84 on *JudgeBench*. Annotations were used only for evaluation and validation and did not modify the datasets, verification assignments, or model outputs.

REPRODUCIBILITY STATEMENT

The work provides the methodological and experimental details needed to reproduce the reported results. The **JDGs** framework, including property extraction, verification assignment, dependency graph construction, executable verification, aggregation, calibration, and the verification-coverage error bound, is described in §3, with assumptions and the proof of Lemma 1 in the main text; dataset descriptions, processing procedures, experimental settings, hyperparameters, baselines, ablations, verification-coverage controls, and sensitivity analyses are reported in §4.

REFERENCES

Shefayat E Shams Adib, Ahmed Alfey Sani, Ekramul Alam Esham, Ajwad Abrar, and Tareque Mohmud Chowdhury. Assessing large language models for medical qa: Zero-shot and llm-as-a-judge evaluation. In *2025 28th International Conference on Computer and Information Technology (ICCIT)*, pp. 3081–3086. IEEE, 2025.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454, 2024.

Miguel Carvalho, Armando J Pinho, and Susana Brás. Resampling approaches to handle class imbalance: a review from a data perspective. *Journal of Big Data*, 12(1):71, 2025.

José Cassola-Bacallao, José Morales-Donaire, Paula Hernández-Montoya, and Brian Keith-Norambuena. Benchmarking llm-as-a-judge models for 5w1h extraction evaluation. *Electronics*, 15(3):659, 2026.

Dominic Chalmers, Richard 'Rick' Hunt, Stella Pachidi, Kristina Potočnik, and David Townsend. The acceleration of artificial intelligence: Rethinking organization and work in an era of rapid technological change. *Journal of Management Studies*, 63(2):285–314, 2026.

Sravan Kumar Chittimalla and Leela Krishna M Potluri. Explainable ai frameworks for large language models in high-stakes decision-making. In *2025 International Conference on Advanced Computing Technologies (ICoACT)*, pp. 1–6. IEEE, 2025.

Nacha Chondamrongkul, Georgi Hristov, and Punnarumol Temdee. Addressing technical challenges in large language model-driven educational software system. *IEEE Access*, 13:12846–12858, 2025.

Emma Croxford, Yanjun Gao, Elliot First, Nicholas Pellegrino, Miranda Schnier, John Caskey, Madeline Oguss, Graham Wills, Guanhua Chen, Dmitriy Dligach, et al. Automating evaluation of ai text generation in healthcare with a large language model (llm)-as-a-judge. *medRxiv*, 2025.

Yuanning Feng, Sinan Wang, Zhengxiang Cheng, Yao Wan, and Dongping Chen. Are we on the right way to assessing llm-as-a-judge? *arXiv preprint arXiv:2512.16041*, 2025.

Apolinar González-Potes, Diego Martínez-Castro, Carlos M Paredes, Alberto Ochoa-Brust, Luis J Mena, Rafael Martínez-Peláez, Vanessa G Félix, and Ramón A Félix-Cuadras. Hybrid ai and llm-enabled agent-based real-time decision support architecture for industrial batch processes: A clean-in-place case study. *AI*, 7(2):51, 2026.

Mohammed Mehedi Hasan, Hao Li, Emad Fallahzadeh, Gopi Krishnan Rajbahadur, Bram Adams, and Ahmed E Hassan. An empirical study of testing practices in open source ai agent frameworks and agentic applications. *Empirical Software Engineering*, 31(5):124, 2026.

Xing Hu, Feifei Niu, Junkai Chen, Xin Zhou, Junwei Zhang, Junda He, Xin Xia, and David Lo. Assessing and advancing benchmarks for evaluating large language models in software engineering tasks. *ACM Transactions on Software Engineering and Methodology*, 2025.

Donghao Huang, Shila Chew, Anna Dutkiewicz, and Zhaoxia Wang. Llm-as-a-judge for scalable test coverage evaluation: Accuracy, operational reliability, and cost. *arXiv preprint arXiv:2512.01232*, 2025.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, 2025a.

Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *ACM transactions on software engineering and methodology*, 34(2):1–23, 2025b.

Xiaoyuan Li, Keqin Bao, Yubo Ma, Moxin Li, Wenjie Wang, Rui Men, Yichang Zhang, Fuli Feng, and Dayiheng Liu. Mtr-bench: A comprehensive benchmark for multi-turn reasoning evaluation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21525–21577, 2026.

Jiaxin Liang, Haotian Miao, Kai Li, Jianheng Tan, Xi Wang, Rui Luo, and Yueqiu Jiang. A review of multi-agent reinforcement learning algorithms. *Electronics*, 14(4):820, 2025.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. In *International Conference on Learning Representations*, volume 2025, pp. 86205–86233, 2025.

Zhiyuan Ma, Jiayu Liu, Xianzhen Luo, Zhenya Huang, Qingfu Zhu, and Wanxiang Che. Advancing tool-augmented large language models via meta-verification and reflection learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2078–2089, 2025.

Xinji Mai, Haotian Xu, Zhong-Zhi Li, Weinong Wang, Jian Hu, Yingying Zhang, Wenqiang Zhang, et al. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving. *arXiv preprint arXiv:2505.07773*, 2025.

Abdul Kadar Muhammad Masum, Md Fokrul Islam Khan, Md Maruf Hassan, Dewan Md Farid, Abu Kowshir Bitto, and Md Arifur Rahman. Multi-model ensemble approach for accurate classification of ocular disorders. In *2025 International Conference on Quantum Photonics, Artificial Intelligence, and Networking (QPAIN)*, pp. 1–6. IEEE, 2025.

Rui Min, Tianyu Pang, Chao Du, Qian Liu, Minhao Cheng, and Min Lin. Improving your model ranking on chatbot arena by vote rigging. *arXiv preprint arXiv:2501.17858*, 2025.

Jie Ruan, Inderjeet Nair, Shuyang Cao, Amy Liu, Sheza Munir, Micah Pollens-Dempsey, Yune-Ting Chiang, Lucy Kates, Nicholas David, Sihan Chen, et al. Expertlongbench: Benchmarking language models on expert-level long-form generation tasks with structured checklists. In *International Conference on Learning Representations*, volume 2026, pp. 33389–33475, 2026.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. Learning to plan & reason for evaluation with thinking-llm-as-a-judge. *arXiv preprint arXiv:2501.18099*, 2025.

Valentin Schmidberger, Manuel Eberhardinger, Setareh Maghsudi, and Johannes Maucher. Generation of programmatic rules for document forgery detection using large language models. In *2025 International Conference on Machine Learning and Applications (ICMLA)*, pp. 1479–1486. IEEE, 2025.

Saeid Sheikhi, Lauri Loven, and Panos Kostakos. Beyond the leaderboard: A survey of the science of evaluation, benchmarking, and methodologies for large language models. *IEEE Access*, 2026.

Evangelia Spiliopoulou, Riccardo Fogliato, Hanna Burnsky, Tamer Soliman, Jie Ma, Graham Horwood, and Miguel Ballesteros. Play favorites: A statistical method to measure self-bias in llm-as-a-judge. *arXiv preprint arXiv:2508.06709*, 2025.

Bian Sun, Zhenjian Wang, Orvill de la Torre, and Zirui Wang. When metrics disagree: Automatic similarity vs. llm-as-a-judge for clinical dialogue evaluation. *arXiv preprint arXiv:2603.00314*, 2026.

Cheng Tan, Dongxin Lyu, Siyuan Li, Zhangyang Gao, Jingxuan Wei, Siqi Ma, Zicheng Liu, and Stan Z Li. Peer review as a multi-turn and long-context dialogue with role-based interactions. *arXiv preprint arXiv:2406.05688*, 2024.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations*, volume 2025, pp. 63277–63303, 2025.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111, 2025.

Xinru Wang, Ming Yin, Eunyee Koh, and Mustafa Doga Dogan. Xagen: An explainability tool for identifying and correcting failures in multi-agent workflows. In *Proceedings of the Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems*, pp. 1–11, 2026a.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *International Conference on Learning Representations*, volume 2024, pp. 43573–43593, 2024.

Yidong Wang, Yunze Song, Tingyuan Zhu, Xuanwang Zhang, Zhuohao Yu, Hao Chen, Chiyu Song, Qiufeng Wang, Zhen Wu, Xin-Yu Dai, et al. Trustjudge: Inconsistencies of llm-as-a-judge and how to alleviate them. In *International Conference on Learning Representations*, volume 2026, pp. 146399–146421, 2026b.

Eric Xue, Ke Chen, Zeyi Huang, Yuyang Ji, and Haohan Wang. Improve: Iterative model pipeline refinement and optimization leveraging llm experts. *arXiv preprint arXiv:2502.18530*, 2025.

Xinyi Yang, Runzhe Zhan, Shu Yang, Junchao Wu, Lidia S Chao, and Derek F Wong. Rethinking prompt-based debiasing in large language model. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26538–26553, 2025.

Zhiyuan You, Xin Cai, Jinjin Gu, Tianfan Xue, and Chao Dong. Teaching large language models to regress accurate image quality scores using score distribution. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14483–14494. IEEE, 2025.

Bowen Zhang, Xianli Liu, Caixu Yue, Shaoyang Liu, Xuebing Li, Steven Y Liang, and Lihui Wang. An imbalanced data learning approach for tool wear monitoring based on data augmentation. *Journal of Intelligent Manufacturing*, 36(1):399–420, 2025.

Weihao Zhang, Yitong Zhou, Huanyu Qu, and Hongyi Li. Loosely-structured software: Engineering context, structure, and evolution entropy in runtime-rewired multi-agent systems. *arXiv preprint arXiv:2603.15690*, 2026.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, volume 2025, pp. 51257–51296, 2025.