# From Many Voices to One: Statistically Principled Aggregation of LLM Judges

**Jitian Zhao***
Department of Statistics
University of Wisconsin-Madison
Madison, WI 53703
jzhao326@wisc.edu

**Changho Shin** *
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53703
cshin23@wisc.edu

**Tzu-Heng Huang**
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53703
thuang273@wisc.edu

**Satya Sai Srinath Namburi GNVV**
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53703
sgnamburi@wisc.edu

**Frederic Sala**
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53703
fsala@wisc.edu

## Abstract

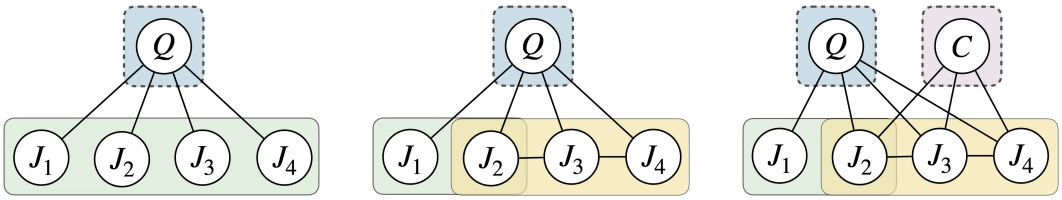
LLM-as-a-judge—often with multiple judges—is now the standard for scalable model evaluation, yet judge biases and correlations can amplify errors. We cast aggregation as inference in a latent-factor Markov random field that jointly models a latent true-quality variable, inter-judge correlations, and confounders (e.g., generation length). We address two key technical challenges—identifiability and learning a higher-rank latent structure—via **CARE**, a two-stage estimator that uses sparse+low-rank structure recovery and tensor decomposition to separate quality from spurious factors. This enables us to better understand the quality and behavior of judges, leading to improved evaluation capabilities. Empirically, it reduces aggregation error by up to **25.15%** and seamlessly incorporates cheaply constructed programmatic judges, while matching or surpassing individual-judge intervention strategies.

## 1 Introduction

Large language models (LLMs) are now widely used for automated evaluation of model outputs. A common practice is to ensemble multiple LLM judges to form consensus scores [1], avoiding the cost of expert annotation [2]. However, such ensembles are unreliable: judges exhibit systematic biases (e.g., verbosity, position) [3, 4, 5], are highly correlated from shared training data, and thus may amplify rather than reduce errors [6, 7]. Existing fixes—including order shuffling, prompt calibration, or fine-tuned evaluators [8, 9, 10, 5]—target individual biases, while aggregation methods like majority vote or averaging [11] rely on unrealistic independence assumptions.

*Equal contribution to this work

39th Conference on Neural Information Processing Systems (NeurIPS 2025).




(a) Naive: only true-quality (b) Connection-aware (c) Confounder-aware (Ours)

Figure 1: **Graphical models for aggregating judge scores under different structural assumptions. (a)** A naive model assumes scores reflect only a true latent quality ($Q$) and that all judges are equally reliable and represent independent views. **(b)** Connection-aware approach models intra-judge interactions ($J_2 - J_3 - J_4$), but still assumes the presence of a single latent quality score. **(c)** Our Confounder-aware model introduces additional latent confounders ($C$) influencing judge scores.

We take a principled approach, recasting multi-judge aggregation as inference in a **higher-rank latent variable Markov Random Field (MRF)**. This model captures (i) a latent quality variable $Q$, (ii) additional confounders (e.g., length, style), and (iii) correlations between judges. Learning such models raises two challenges: (1) estimating parameters without observing latent factors, and (2) identifying which factor corresponds to true quality rather than spurious signals. Our solution, **CARE**, combines sparse+low-rank decomposition with a tensor step to separate $Q$ from confounders, and further supports integration of *programmatic judges*—cheaply synthesized evaluation functions that expand the judge pool [12].

**Our Contributions.** CARE (i) introduces a confounder-aware aggregation framework unifying single-judge debiasing with principled statistical fusion, (ii) provides identifiability guarantees, sample complexity bounds, and misspecification analysis, (iii) reduces aggregation error by up to **25.15%** on public benchmarks compared to majority vote, weak supervision baselines, and prompt-level interventions, and (iv) seamlessly integrates programmatic judges while supporting progressive expansion of evaluator pools.

## 2 CARE: Confounder-Aware Aggregation for Reliable Evaluation

**Motivation.** LLM-as-a-judge is appealing for scalable evaluation, but naive ensembling can amplify shared biases. Judges may favor longer generations, prefer particular styles, or otherwise correlate in ways that obscure the underlying quality. Existing heuristics (e.g., weighting or filtering judges) only partially address these issues. We instead adopt a probabilistic graphical model perspective, which provides a principled way to separate latent true quality from other spurious factors.

**Latent-factor MRF.** We model judges as nodes in a Markov random field with multiple latent factors: one unknown true-quality variable $Q$, one or more confounders $C$, and the observed judge scores $X_1, \ldots, X_m$. This higher-rank structure captures both genuine signal and correlated biases (Fig. 3). The main technical challenges are (i) identifying which latent dimension corresponds to $Q$, and (ii) estimating such higher-rank latent-variable models from limited samples.

**CARE algorithm.** Our approach, CARE (Confounder-Aware Aggregation), addresses these challenges with a two-stage estimator. First, we recover a sparse+low-rank decomposition of the precision matrix of judge scores: the sparse part captures direct conditional links between judges, while the low-rank part reveals latent factors. Second, we apply a symmetric tensor decomposition to resolve rotational ambiguity and isolate $Q$ from confounders. This produces an interpretable set of factor loadings, showing how each judge aligns with true quality versus spurious dimensions. CARE then aggregates scores along the inferred quality dimension, yielding robustness to confounders and correlations. Programmatic judges, such as length counters or keyword checkers, can be included alongside LLM judges and are naturally placed onto confounder dimensions when appropriate. Full pseudocode, optimization details, and proofs appear in Appendix C.

**Theoretical guarantees.** We provide three main results for Algorithm 1. (i) *Identifiability:* under latent-independence and orthogonality assumptions, CARE exactly recovers latent directions and is stable to mild perturbations (App. D.2). (ii) *Sample complexity:* we bound the number of samples needed for consistent estimation of latent-observable connections, with rates depending on eigengaps

**Algorithm 1** CARE (Confounder-Aware Aggregation) (Condensed, full version in Algorithm 2)

**Input:** Judge score matrix $J \in \mathbb{R}^{n \times p}$
**Output:** Estimated true-quality scores $\{\hat{q}^{(i)}\}_{i=1}^{n}$

1: Estimate judge graph sparse structure from $J$
2: Recover sparse + low-rank decomposition $(\hat{S}, \hat{L})$
3: Extract latent factors from $\hat{L}$ (SVD or tensor methods; see App. C)
4: Identify the quality factor among latent components
5: Aggregate along this factor to produce $\hat{q}^{(i)}$

Table 1: Aggregation performance across different datasets, measured by MAE and Kendall's $\tau$. CARE outperforms baseline methods in most cases.

| | **FeedbackQA** | | **HelpSteer2** | | **UltraFeedback** | |
|---|---|---|---|---|---|---|
| | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) |
| MV | 0.8812 | 0.3703 | 0.9951 | 0.1629 | 0.8522 | 0.2985 |
| AVG | 0.8492 | 0.4497 | 0.9822 | 0.1611 | 0.6860 | 0.3621 |
| WS | 0.8144 | 0.4401 | 1.3030 | 0.1511 | 1.1603 | 0.3306 |
| UWS | 0.9051 | **0.4580** | 0.9849 | 0.1697 | 0.6794 | 0.3669 |
| CARE | **0.7866** | 0.4542 | **0.9742** | **0.1805** | **0.6379** | **0.3806** |

and manifold curvature (App. D.3). (iii) *Misspecification error:* omitting confounders introduces systematic bias; we provide explicit bounds on the resulting conditional-mean errors (App. D.4).

## 3 Experimental Results

We evaluate CARE across diverse experimental setups, including real-world and semi-synthetic datasets, to validate the following key claims:

- **Improving aggregation of LLM judges:** CARE produces more accurate and robust aggregate scores from multiple LLM judges compared to existing methods (Sec. 3.1).
- **Effective integration of programmatic judges:** CARE can integrate programmatic judges, which are often systematically biased, by explicitly modeling confounders (Sec. 3.2).
- **Progressive expansion of judges:** CARE robustly incorporates additional judges over time, adapting to larger evaluation pools (Sec. 3.3).
- **Competitiveness against manual interventions:** CARE matches or surpasses prompt-level interventions at the individual judge level, avoiding costly manual tuning (Sec. 3.4).

Additional experiments, including robustness under controlled confounding factors, and validation of theoretical results in synthetic settings, are deferred to Appendix E. We also offer ablations, per-judge breakdowns, and further programmatic judge analyses in Appendix E.

### 3.1 Improving Aggregation of LLM judges

**Setup.** We compare aggregation methods using the 10 LLM judges (listed in the Appendix E). To ensure consistency, we adapt the prompt template from [13], modifying it to fit our experimental setup. The exact used prompt is provided in Appendix E.

**Results.** We report aggregation performance in Table 1. CARE consistently outperforms baseline methods, achieving the lowest MAE on FeedbackQA (0.7866) and UltraFeedback (0.6379), surpassing majority vote (MV) by **10.74%** and **25.15%**, respectively. These gains demonstrate CARE's ability to model correlations among LLM judges and mitigate compounding biases.

### 3.2 Effective Integration of Programmatic Judges

**Setup.** We integrate our LLM-based evaluators with ten programmatic judges, each encoding its evaluation logic into executable code synthesized by OpenAI's GPT-4o [14]. These judges are designed to assess response quality through specific, individual dimension, such as *structure,*

Table 2: Performance on different datasets using both LLM and programmatic judges. Programmatic judges are beneficial in FeedbackQA but may introduce noise in HelpSteer2 and UltraFeedback. In both cases, CARE consistently outperforms other baselines.

| | **FeedbackQA** | | **HelpSteer2** | | **UltraFeedback** | |
|---|---|---|---|---|---|---|
| | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) |
| MV | 0.8607 | 0.3815 | 1.0244 | **0.1465** | 0.8751 | 0.3179 |
| AVG | 0.8128 | 0.4671 | 1.1012 | 0.1268 | 1.0371 | **0.3733** |
| UWS | 0.8179 | **0.4816** | 0.9992 | 0.1040 | 0.9534 | 0.3047 |
| CARE | **0.7582** | 0.4796 | **0.9800** | 0.1398 | **0.7351** | 0.3520 |

Table 3: Comparison of aggregation methods using individually intervened LLM judges. While other baselines aggregate scores from debiased LLM judges, CARE operates directly on raw outputs.

| | **FeedbackQA** | | **HelpSteer2** | | **UltraFeedback** | |
|---|---|---|---|---|---|---|
| | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) |
| MV | 0.8004 | 0.3964 | 0.9951 | 0.1629 | 0.8562 | 0.2799 |
| AVG | 0.8029 | 0.4412 | 0.9822 | 0.1611 | 0.6801 | 0.3704 |
| WS | **0.7674** | 0.4429 | 1.3030 | 0.1511 | 1.1516 | 0.3588 |
| UWS | 0.8117 | 0.4390 | 0.9849 | 0.1697 | 0.6683 | 0.3782 |
| CARE | 0.7866 | **0.4542** | **0.9742** | **0.1805** | **0.6379** | **0.3806** |

*readability, safety, relevance, and factuality*. While it's cost-effective to construct them, their deterministic nature may introduce systematic biases, potentially leading to noisy evaluation signals. This setup allows us to test CARE's robustness in a more challenging aggregation scenario. Further details on the programmatic judge generation process are provided in Appendix E.

**Results.** Table 2 shows that adding programmatic judges improves FeedbackQA, where CARE attains the lowest MAE (0.7582) and highest $\tau$ (0.4796), outperforming MV by **11.92%**. On HelpSteer2 and UltraFeedback, performance drops (MAE 0.9800 and 0.7351) but still surpasses MV by **4.33%** and **15.99%**. Overall, CARE consistently outperforms baselines on MAE, even under noisy signals.

### 3.3 Progressive Judge Expansion

**Setup.** Next, we start with a fixed set of LLM judges and progressively add programmatic judges from a pool of 23. At each step, we greedily select the programmatic judge that yields the largest improvement in the validation of MAE. The process stops when no further reduction in validation MAE is observed. We evaluate aggregation methods as in previous sections, using FeedbackQA, where programmatic judges were most beneficial.

**Results.** Figure 2 reports the results of scaling the number of programmatic judges. As the number of programmatic judges increases, CARE consistently achieves lower error, demonstrating its ability to adapt and improve with additional supervision. These findings suggest a promising path toward building dynamic and expandable judge ensembles.

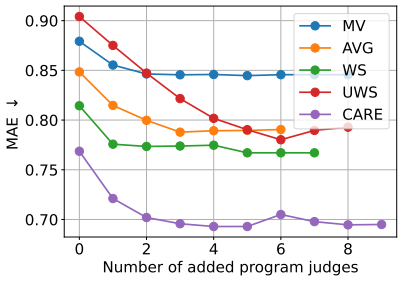



Figure 2: Progressive judge selection on the FeedbackQA dataset. CARE robustly integrates new judges and consistently outperforms baseline aggregation methods.

### 3.4 Comparison with Individual Intervention

**Setup.** An alternative to our confounder-aware approach is direct interventions at the individual judge level. Specifically, we compare CARE to prompt-based interventions proposed by [15], which instruct LLM judges to account for known sources of bias. The intervened prompt used for this comparison is included in Appendix E.

**Results.** Table 3 presents the results. While bias-aware prompting improves performance in most cases, CARE remains the top performer in the majority of settings, and even when not, it is com-

petitive with the best. This suggests that CARE can effectively mitigate biases without relying on careful prompt engineering.

We next evaluate robustness under controlled bias injections (e.g., beauty, authority, gender), as well as synthetic experiments validating our theory. Due to space constraints, these results are presented in Appendix E, where we also include expanded analyses such as prompt-based intervention effects (Appendix E.5) and controlled confounder demonstrations (Appendix E.6, E.7).

## 4 Conclusion

We presented CARE, a confounder-aware aggregation framework that casts multi-judge scoring as inference in a higher-rank latent-variable model. It explicitly models shared confounders, provides principled estimators with identifiability guarantees, and achieves consistent gains on public benchmarks, reducing MAE and improving Kendall's $\tau$ by up to **25.15%**.

## References


[1] Zhengyu Hu, Jieyu Zhang, Zhihan Xiong, Alexander Ratner, Hui Xiong, and Ranjay Krishna. Language model preference evaluation with multiple weak evaluators. *arXiv preprint arXiv:2410.12869*, 2024.

[2] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[3] Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biase in LLM-as-a-judge. In *The Thirteenth International Conference on Learning Representations*, 2025.

[4] Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic investigation of position bias in pairwise comparative assessments by llms. *arXiv preprint arXiv:2406.07791*, 2024.

[5] 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 *The Twelfth International Conference on Learning Representations*.

[6] Daniel Deutsch, Rotem Dror, and Dan Roth. On the limitations of reference-free evaluations of generated text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10960–10977, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[7] Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in llm-as-a-judge. 2025.

[8] Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or LLMs as the judge? a study on judgement bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8301–8327, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[9] Haitao Li, Junjie Chen, Qingyao Ai, Zhumin Chu, Yujia Zhou, Qian Dong, and Yiqun Liu. Calibraeval: Calibrating prediction distribution to mitigate selection bias in llms-as-judges. *arXiv preprint arXiv:2410.15393*, 2024.

[10] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. In *The Thirteenth International Conference on Learning Representations*.

[11] Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024.

[12] Tzu-Heng Huang, Harit Vishwakarma, and Frederic Sala. Time to impeach llm-as-a-judge: Programs are the future of evaluation. *arXiv preprint arXiv:2506.10403*, 2025.

[13] Aymeric Roucher. Using LLM-as-a-judge for an automated and versatile evaluation. `https://huggingface.co/learn/cookbook/en/llm_judge`, n.d. Accessed: 2025-05-15.

[14] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[15] Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, et al. Justice or prejudice? quantifying biases in llm-as-a-judge. In *Neurips Safe Generative AI Workshop 2024*.

[16] Tom Kocmi and Christian Federmann. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 193–203, 2023.

[17] Chenhui Shen, Liying Cheng, Xuan-Phi Nguyen, Yang You, and Lidong Bing. Large language models are not yet human-level evaluators for abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4215–4233, 2023.

[18] Cheng-Han Chiang and Hung yi Lee. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 15607–15631, 2023.

[19] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9440–9450, 2024.

[20] Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

[21] Junsoo Park, Seungyeon Jwa, Meiying Ren, Daeyoung Kim, and Sanghyuk Choi. OffsetBias: Leveraging debiased data for tuning evaluators. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1043–1067, 2024.

[22] Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Shuai Wang, Cuiyun Gao, and Yang Liu. Split and merge: Aligning position biases in LLM-based evaluators. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11084–11108, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[23] Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rockt"aschel, and Ethan Perez. Debating with more persuasive LLMs leads to more truthful answers. *arXiv preprint arXiv:2402.06782*, 2024.

[24] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023.

[25] Ruosen Li, Teerth Patel, and Xinya Du. PRD: Peer rank and discussion improve large language model based evaluations. *Transactions on Machine Learning Research*, 2024.

[26] Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment*, volume 11, pages 269–282, 2017.

[27] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, pages 362–375, 2019.

[28] Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020.

[29] Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Carl Roberts, and Frederic Sala. Universalizing weak supervision. In *International Conference on Learning Representations (ICLR)*, 2022.

[30] Sebastian Rühling Cachay, Benjamin Boecking, and Artur Dubrawski. End-to-end weak supervision. In *Advances in Neural Information Processing Systems*, 2021.

[31] Zheng Kuang, Chidubem Arachie, Brian Liang, Pratyush Narayana, Grace DeSalvo, Michael Quinn, Bo Huang, Gabriel Downs, and Yiming Yang. Firebolt: Weak supervision under weaker assumptions. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, 2022.

[32] Changho Shin, Sonia Cromp, Dyah Adila, and Frederic Sala. Mitigating source bias for fairer weak supervision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[33] Pat Verga, Sebastian Hofstätter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, et al. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024.

[34] Zhengyu Hu, Jieyu Zhang, Zhihan Xiong, Alexander Ratner, Hui Xiong, and Ranjay Krishna. Language model preference evaluation with multiple weak evaluators. *arXiv preprint arXiv:2410.12869*, 2024.

[35] Hossein A. Rahmani, Emine Yilmaz, Nick Craswell, and Bhaskar Mitra. Judgeblender: Ensembling judgments for automatic relevance assessment. *arXiv preprint arXiv:2412.13268*, 2024.

[36] Venkat Chandrasekaran, Pablo A. Parrilo, and Alan S. Willsky. Latent variable graphical model selection via convex optimization. *The Annals of Statistics*, 40(4), August 2012.

[37] Animashree Anandkumar, Rong Ge, Daniel J Hsu, Sham M Kakade, Matus Telgarsky, et al. Tensor decompositions for learning latent variable models. *J. Mach. Learn. Res.*, 15(1):2773–2832, 2014.

[38] Yi Yu, Tengyao Wang, and Richard J Samworth. A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

[39] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, pages 362–375, 2019.

[40] Daniel Hsu and Sham M Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20, 2013.

[41] Zichao Li, Prakhar Sharma, Xing Han Lu, Jackie Chi Kit Cheung, and Siva Reddy. Using interactive feedback to improve the accuracy and explainability of question answering systems post-deployment. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 926–937, 2022.

[42] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. 2023.

[43] Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*, 2024.

[44] Maurice Kendall. A new measure of rank correlation. *Biometrika*, pages 81–89, 1938.

[45] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[46] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[47] Qwen Team. Qwen3, April 2025.

[48] Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

[49] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[50] Tzu-Heng Huang, Catherine Cao, Vaishnavi Bhargava, and Frederic Sala. The alchemist: Automated labeling 500x cheaper than llm data annotators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[51] Tzu-Heng Huang, Catherine Cao, Spencer Schoenberg, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. Scriptoriumws: A code generation assistant for weak supervision. *arXiv preprint arXiv:2502.12366*, 2025.

[52] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 491–500, 2019.

The appendix is structured as follows. It starts with the glossary table, defining key notations used throughout the paper in Appendix A. Next, Appendix B discusses additional related work. In Appendix C, we introduce details about our tensor-based CARE algorithm, discussion for general CARE method, and additional discussion about method heuristics. Following this, Appendix D offers theoretical support of our approach and supported proofs. It includes the graphical model formulation, graph structure recovery error bound, sample complexity, and the misspecification error arising from incorrectly characterized confounding factors. Subsequently, Appendix E provides experimental details and additional experiment results. Finally, Appendix F concludes by discussing the broader impacts and limitations of the work.

# A Glossary

The notations are summarized in Table 4 below.

Table 4: Glossary of variables and symbols used in this paper.

| Symbol | Definition |
|---|---|
| $(J_1, \ldots, J_p)$ | $p$ vector of Judges score |
| $Q$ | True-quality latent variable |
| $(C_1, \ldots, C_k)$ | $k$ latent confounder variables ($\mathbb{R}^p$) |
| $H$ | All the hidden variables (true + confounder) i.e ($Q\ C_1, \ldots, C_k$) |
| $h$ | dimension of $H$ i.e all hidden variables = $k$ + 1 |
| $X$ | Observed judge score matrix of dimension ($n \times p$) where $n$ is the number of examples and $p$ is the number of judges |
| $K$ | Precision matrix |
| $K_{JJ}$ | Observable-observable connection matrix |
| $K_{JH}$ | Observable-latent connection matrix |
| $K_{HH}$ | Latent-latent connection matrix |
| $\Sigma_{JJ}$ | Covariance matrix of observable variables |
| $S$ | Sparse matrix ($\mathbb{R}^{p\times p}$) which encodes edges between judges |
| $L$ | Low-rank matrix (with $rank(L) \leq h$) which captures dependencies mediated by latent variables |
| $R$ | Rotation matrix ($\mathbb{R}^{h\times h}$) |
| $\gamma_n$ | Regularization for sparse and low-rank matrix $S$ in Algorithm 2 |
| $\tau$ | Regularization for low-rank matrix $L$ in Algorithm 2 |
| $\hat{s}_{\text{agg}}^{(i)}$ | Aggregated scores for $i$th example in the dataset from $p$ judges |
| $\hat{\Sigma}$ | Sample precision estimation or covariance matrix |
| $\hat{S}$ | Sample Sparse matrix ($\mathbb{R}^{p\times p}$) which encodes direct connectional edges among judges |
| $\hat{L}$ | Sample Low-rank matrix (with $rank(L) \leq h$) which captures dependencies mediated by latent variables |
| $U$ | Latent factor extraction matrix i.e latent-judge connections ($\mathbb{R}^{p\times h}$) from Algorithm 2 |
| $\Theta$ | Precision matrix |
| $w$ | Weight for aggregating judges |
| $\lambda$ | Singular values of $L$ |
| $u^\star$ | Singular vector of $L$ corresponds to true quality factor |
| $\lambda^\star$ | Singular value of $L$ that corresponds to true quality factor |
| $\mu_{qc}$ | Conditional mean of judges given $Q = q, C = c$ |
| $\hat{\mu}_{qc}$ | Estimated conditional mean of judges given $Q = q, C = c$ |
| $\pi_{qc}$ | Probability of $Q = q, C = c$ |
| $\hat{\pi}_{qc}$ | Estimation of probability of $Q = q, C = c$ |
| $\{\mathcal{G}_\ell\}_{\ell=1}^3$ | Groups of judges that are independent of judges outside the group |
| $\hat{T}$ | Empirical 3-way tensor |
| $\hat{\mu}_{qc}^{(1)}, \hat{\mu}_{qc}^{(2)}, \hat{\mu}_{qc}^{(3)}$ | Estimated conditional mean of three views |
| $\hat{\mu}_{\rho(r)}$ | Estimated conditional mean of judges after permutation |
| $\mu_{\text{anchor}(r)}$ | Conditional mean of anchor sets |

# B Related Work

## B.1 Biases in LLM–as–a–Judge

Large language models have quickly become the standard automatic evaluators for generation tasks because they correlate well with human judgments in translation and summarization [16, 17, 18]. Yet a growing body of work shows that these models are far from impartial. **Positional bias**—preferring the *second* answer in a pairwise comparison—was first noted in MT-Bench [2] and later quantified in detail by [19], who observed reversals of up to 30% when simply swapping order. **Verbosity bias**, wherein longer answers receive higher scores regardless of quality, is highlighted by [8]. LLM judges also display **self-enhancement bias**, overrating responses produced by models from the same family [20]. Less studied but equally problematic are **concreteness/authority biases**: judges over-reward answers that contain citations, numbers, or confident tone even when these features are irrelevant [21].

Mitigation strategies span two levels. *Prompt-level interventions* randomize answer order, enforce symmetric formatting, and instruct the judge to ignore superficial features [19, 22]. Adding chain-of-thought rationales or decomposing the rubric into sub-criteria (accuracy, conciseness, style) also moderates shallow heuristics [23]. On the *model level*, fine-tuned evaluators such as JudgeLM [24] and Split-and-Merge Judge [22] are trained on curated data that explicitly counter positional and length biases. Peer-review and debate schemes go a step further: PRD lets a second LLM critique the first judge and often corrects biased decisions [25], while [23] show that dialog with a more persuasive model leads to more truthful verdicts.

Despite progress, most debiasing work treats a *single* judge in isolation. When evaluations aggregate many LLM scorers—for robustness, cost sharing, or diversity—biases can compound in complex ways that individual fixes do not capture.

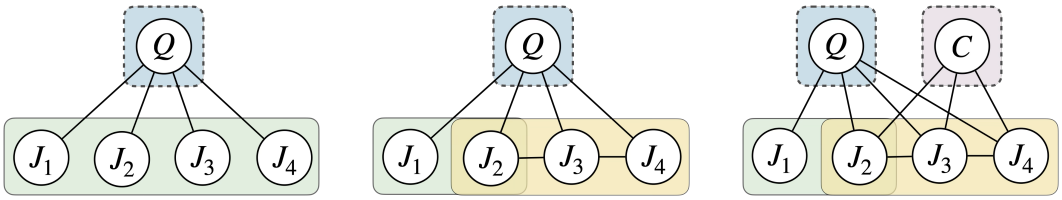
## B.2 Label Aggregation for Multiple Noisy Evaluators

**Weak-supervision.** Treating each LLM prompt or model as a noisy *labeling function* aligns aggregation with modern weak supervision. Snorkel [26, 27] estimates source accuracies and dependencies to denoise programmatic labels, laying the foundation for LLM-prompt aggregation. [28] introduces a scalable moment-matching estimator with closed-form weights.[29] generalizes label models beyond categorical labels to arbitrary metric spaces, greatly expanding their applicability. [30] jointly optimizes a classifier and a differentiable label model, outperforming two-stage pipelines when sources are dependent. Firebolt further removes requirements on known class priors or source independence, estimating class-specific accuracies and correlations in closed form [31]. [32] shows that fixing source bias in labeling functions using optimal transport can improve both accuracy and fairness.

**Aggregation of multiple *LLM* judges.** Recent work shows that *ensembling smaller evaluators can beat a single large judge*. The **PoLL** jury combines three diverse 7–35B models and attains higher correlation with human ratings than GPT-4 while costing 7× less and reducing bias [33]. **GED** merges preference graphs from weak evaluators (Llama3-8B, Mistral-7B, Qwen2-7B) and denoises cycles; its DAG ranking surpasses a single 72B judge on ten benchmarks [34]. **JudgeBlender** ensembles either multiple models or multiple prompts, improving precision and consistency of relevance judgments over any individual LLM [35]. These findings echo classic "wisdom-of-crowds" results—when paired with principled aggregation, a panel of smaller, heterogeneous judges can outperform a much larger model, offering a practical path toward reliable, low-cost evaluation.

## B.3 Our Contribution in Context

Prior research either (i) debiases one judge at a time or (ii) aggregates multiple judges assuming independent noise. Our confounder-aware aggregation unifies these threads. We posit latent factors (e.g., verbosity, formality) that influence *all* judges simultaneously and show how to infer both the latent truth and the shared confounders. This yields more reliable consensus scores when individual judges—human or LLM—share systemic biases.




(a) Naive: only true-quality (b) Connection-aware (c) Confounder-aware (Ours)

Figure 3: **Graphical models for aggregating judge scores under different structural assumptions. (a)** A naive model assumes scores reflect only a true latent quality ($Q$) and that all judges are equally reliable and represent independent views. **(b)** Connection-aware approach models intra-judge interactions ($J_2 - J_3 - J_4$), but still assumes the presence of a single latent quality score. **(c)** Our Confounder-aware model introduces additional latent confounders ($C$) influencing judge scores.

# C Algorithm Details

We introduce CARE (Confounder-Aware Aggregation for Reliable Evaluation), a graphical model–based framework that robustly estimates the true quality of LLM-as-a-judge assessments by explicitly modeling the influence of both a latent true-quality variable and additional latent confounders on observed judge scores. This section details the implementation of CARE, including the full CARE tensor algorithm, an SVD baseline for comparison, generalizations beyond Gaussian assumptions, practical heuristics for symmetry breaking and handling non-orthogonal latent factors, and justification for sparse structure recovery in mixed Gaussian data.

## C.1 Graphical Model Framework And Assumptions

For each prompt-response pair, we observe scores $J = (J_1, \ldots, J_p)^\top$ from $p$ judges. We assume these observed scores depend on latent variables including one *true quality variable* $Q$ and one or more *confounders* $C = (C_1, \ldots, C_k)$, which we define as $H = (Q, C)$. Our graphical model encodes the conditional independence structure among the nodes in $(J, Q, C)$: if there is no edge between a pair of nodes, they are independent conditioned on the other nodes. An example is shown on the right in Fig. 3. We assume this structure is *sparse*; i.e., there are not too many edges in the graph, and make this precise later on.

This framework is quite general and is compatible with a variety of distributions. For example, we may take $J, Q, C$ to involve discrete variables, Gaussians, or mixed models. We can take the model to be an MRF or alternatively a mixture model. Our approaches are compatible with a broad range of choices, with practitioners able to select the most suitable modeling assumptions for their settings.

**Goals and Assumptions.** Under the chosen modeling assumptions, our goal is to learn the distribution over $J, Q, C$. This involves handling ***three challenges***. First, **C1**: *we never observe the latents in* $H$—neither ground truth nor confounders. Second, **C2**: we *cannot assume any particular interaction* in the graph. Third, **C3**: even if we recover the model parameters, we must be able to distinguish between $Q$ and the confounders $C$ *to identify the model*. The latter is required to discover ***which latent is the ground-truth quality—and which is a confounder***. Once these obstacles are overcome, we seek to perform aggregation, e.g., compute a posterior $P(Q|J)$, the Bayesian estimate for the latent true quality conditioned on all observable judge scores.

In the following, we will work under the assumption that the judge scores $J$ conditioned on the latents form a multivariate Gaussian distribution, i.e., $J \mid H \sim \mathcal{N}(\mu_H, \Sigma)$, where $\mu_H$ is the conditional mean of observable variables. We defer other scenarios to the Appendix.

## C.2 CARE Algorithm

The idea behind CARE is to examine two techniques, each of which is stymied by one of the obstacles **C2** or **C3** and to *delicately combine them in a novel way*. First, the sparsity of the conditional independence graph is encoded into an two-dimensional object that can be empirically estimated (e.g., the observable covariance matrix, or a cross-moment matrix). However, the presence of the latent variables (**C1**) obscures this structure—but a *sparse* + *low-rank decomposition* can

**Algorithm 2** CARE: Confounder-Aware Aggregation for Reliable Evaluation

**Input:** Score matrix $J \in \mathbb{R}^{n \times p}$, parameters $(\gamma_n, \tau)$, decomposition method $\mathcal{D} \in \{\text{SVD}, \text{Tensor}\}$
**Output:** Estimated True Quality $\{\hat{q}^{(i)}\}_{i=1}^{n}$

1: **Graph Sparse Structure Estimation:** Compute appropriate observed matrix $f(J)$.
2: **Sparse + low-rank decomposition:**

$$(\hat{S}, \hat{L}) \leftarrow \arg\min_{S,L} \ \tfrac{1}{2}\|f(J) - S - L\|_F^2 + \gamma_n(\|S\|_1 + \tau\|L\|_*)$$

3: **Latent Factor Extraction:**
4: **if** $\mathcal{D} = \text{SVD}$ **then** ▷ Fully Gaussian scenario
5: Compute $U\Lambda U^\top \leftarrow \text{SVD}(\hat{L})$, where $U \in \mathbb{R}^{p \times h}$
6: **else if** $\mathcal{D} = \text{Tensor}$ **then** ▷ Binary-Gaussian mixture scenario
7: Partition judges into independent groups using $\hat{S}$
8: Form empirical third-order tensor from judge groups
9: Run tensor decomposition, obtain latent conditional means $\mu_{qc}$ and mixture proportions $\pi_{qc}$
10: **end if**
11: **Symmetry Breaking:** Identify the true-quality factor using heuristics described in §C.3
12: **Latent Quality Estimation:** Use the identified quality factor, compute $\hat{q}^{(i)}$ for each example, where $\hat{q}^{(i)} = P(Q = 1 \mid J_i)$ for mixture model or $\hat{q}^{(i)} = \mathbb{E}[Q \mid J]$ for fully gaussian

reveal it [36]. However, while we can decompose the resulting low-rank term via SVD in the hope of identifying the model, we can only do so *up to rotations*. Therefore we are blocked by **C3**.

Conversely, tensor product decompositions [37] exploit tensor rigidity to enable this decomposition to be uniquely identified. However, for these techniques the judges must be independent conditioned on the latents—and we cannot assume this by **C2**.

CARE (Algorithm 2) combines these approaches. First, it estimates the underlying graph structure from the observed judge scores via the sparse + low-rank decomposition, overcoming **C1** and **C2**. It then uses recovered sparse term to estimate the graph and discover subsets of judges with sufficient conditional independence. These sets are then used to construct a tensor that can be decomposed via standard approaches (e.g., tensor power method) to recover the model, mitigating **C3**.

This procedure is then followed by a symmetry-breaking step. This requires a weak assumption on the quality of the judges; in practice, even this assumption can be removed by employing simple heuristics to identify the true-quality factor among the latent factors. Finally, we aggregate judge scores into robust evaluations by weighting according to loadings from the identified quality factor.

We study two special cases to build our intuition; more general settings are shown in the Appendix.

**CARE For Gaussian Mixtures.** We have binary latents $(Q, C)$ with $\Pr\big(Q = q,\, C = c\big) = \pi_{qc}$, where the judges follow a Gaussian conditional distribution with mean $\mu_{qc} \in \mathbb{R}^p$ and covariance $\Sigma$:

$$J \;\big|\; (Q = q,\, C = c) \;\sim\; \mathcal{N}\big(\mu_{qc},\, \Sigma\big), \qquad (q, c) \in \{0, 1\}^2.$$

Here, performing the sparse + low-rank decomposition and obtaining $\hat{L}$ is insufficient: the eigen-decomposition of $\hat{L}$ does not directly yield identifiable latent-judge connections. We rely on third-order tensor statistics to identify conditional distributions explicitly:

$$\mathbb{E}(X_1 \otimes X_2 \otimes X_3 \mid Q, C) = \mathbb{E}(X_1 \mid Q, C) \otimes \mathbb{E}(X_2 \mid Q, C) \otimes \mathbb{E}(X_3 \mid Q, C),$$

where judges are partitioned into independent groups $X_1, X_2, X_3$ using the learned sparse structure $\hat{S}$. Performing a tensor decomposition yields the conditional means $\mu_{qc}$ and mixture proportions $\pi_{qc}$. Then, applying Bayes' rule allows estimation of latent variables given observed scores:

$$P(Q = 1|J) \propto \pi_{10}\mu_{10} + \pi_{11}\mu_{11}. \tag{1}$$

**CARE for Fully Gaussian Models.** Under the fully Gaussian assumption, latent variables $H$ are continuous, and the inverse covariance matrix (the *precision* matrix) encodes independence:

$$\Sigma = \text{Cov}\big[(J, H)^\top\big], \quad \Sigma^{-1} = K = \begin{pmatrix} K_{JJ} & K_{JH} \\ K_{HJ} & K_{HH} \end{pmatrix}, \quad S = K_{JJ}, \quad L = K_{JH}K_{HH}^{-1}K_{HJ}.$$

If assuming connections $K_{JH}$ between latent variables and judges are orthogonal and no direct connections among latent variables (i.e. $K_{HH}$ is diagonal), the low-rank matrix $\hat{L}$ admits eigen-decomposition $\hat{L} = U\Lambda U^\top$, where eigenvectors in $U$ directly correspond to latent-judge edges ($K_{JH}$), and eigenvalues correspond to $K_{HH}$. Each eigenvector represents how one latent variable influences observable judges. With these edges recovered, the conditional mean of true quality $Q$ can be estimated by $\mathbb{E}(Q \mid J) = K_{QQ}^{-1} K_{QJ} J$, a weighted linear combination of observed scores.

The fully Gaussian model prevents decomposing the low-rank term uniquely (due to rotational invariance). This holds regardless of whether we apply SVD or a tensor decomposition, leading to the special handling in Algorithm 2. As a result, in this case, orthogonal and independent latent assumptions are needed for identifying the latent-judge connection. This works the best when each judge is connected to exactly one latent variable. If a judge depends on *both* the confounder $C$ and the true quality $Q$ with comparable weights, the recovered columns $\{\hat{\mu}_r\}$ are only identifiable up to an arbitrary rotation, causing estimation errors.

### C.3 Heuristics for Identifiability and Robust Estimation

Any instantiation of CARE will require symmetry-breaking procedures for latent variable identifiability. For example, the fully Gaussian case needs a heuristic to identify the true-quality direction among latent factors, distinguishing $Q$ from confounders $C$. In the binary-Gaussian mixture scenario, an additional step resolves ambiguity between latent states ($Q = 0$ vs. $Q = 1$). Doing so will require additional information that can come from modeling assumptions, the use of ground-truth samples, or heuristics. We detail some examples below:

**Identifying True-Quality Factor for Joint-Gaussian Model.** We introduce heuristics particularly aimed at distinguishing the true-quality latent variable from confounding latent variables. First, the *human-anchor criterion* leverages a small validation set containing human ratings. By including these human judgments in the graphical model, we anchor the latent quality variable to ground truth by selecting the latent factor exhibiting the strongest connection to the human evaluations. Second, we apply a *loading balance heuristic*, identifying the true-quality factor as one that loads broadly and with similar magnitude across all competent judges. Conversely, factors dominated by a few judges typically indicate shared confounding rather than true quality.

**Identifying Latent States for Mixed Model.** In scenarios such as the tensor-based method, symmetry breaking additionally involves distinguishing latent states corresponding to different quality levels (e.g., $Q = 0$ versus $Q = 1$). In practice, we can use known labeled samples (such as high-quality examples) to anchor and identify latent-state configurations. By comparing different latent configurations with these known labeled samples, we select the latent-state assignment that best aligns with empirical observations, effectively removing latent state ambiguity.

### C.4 SVD Baseline in Synthetic Experiment

We form the empirical two-way moment between view 1 and view 2:

$$\widehat{M}_{1,2} \;=\; \frac{1}{n}\sum_{i=1}^{n} X_1^{(i)}\, X_2^{(i)\,\top} \;=\; \sum_{q,c} \pi_{q,c}\, \mu_{1,q,c}\, \mu_{2,q,c}^{\top} \;+\; \text{sampling noise},$$

where $\pi_{q,c} = \Pr[Q = q, C = c]$ and $\mu_{v,q,c} = E[J_v \mid Q = q, C = c]$ for judge/view $v$ A singular-value decomposition

$$\widehat{M}_{1,2} \;=\; U_{12}\, \Sigma_{12}\, V_{12}^{\top}$$

yields factor matrices

$$U_{12}\, \Sigma_{12}^{1/2} \approx [\mu_{1,q,c}]\, R, \quad V_{12}\, \Sigma_{12}^{1/2} \approx [\mu_{2,q,c}]\, R,$$

where $R \in O(4)$ is an unknown orthogonal matrix.

Repeating on $\widehat{M}_{1,3} = \frac{1}{n}\sum_i X_1^{(i)} X_3^{(i)\,\top} = U_{13}\, \Sigma_{13}\, V_{13}^{\top}$ produces a second rotated copy of $[\mu_{1,q,c}]$. We solve the Procrustes problem

$$R \;=\; \arg\min_{O \in O(4)} \left\| U_{12}\, \Sigma_{12}^{1/2} \;-\; U_{13}\, \Sigma_{13}^{1/2}\, O \right\| * F,$$

**Algorithm 3** CARE (T)

**Input:** Score matrix $J \in \mathbb{R}^{n\times p}$, tolerance $\varepsilon$.
**Output:** Estimates $\{\hat{\mu}_{qc}, \hat{\pi}_{qc}\}_{q,c\in\{0,1\}}$.

**A. Anchor discovery (graph partition)**

Compute the sample covariance $\hat{\Sigma} = J^\top J/n$ and perform the sparse+low-rank split $\hat{\Sigma} \approx \hat{S} + \hat{L}$ (Alg. 2).

Partition judges into three disjoint groups $\{\mathcal{G}_\ell\}_{\ell=1}^3$ that satisfy

$$a \neq b,\ j_1 \in \mathcal{G}_a,\ j_2 \in \mathcal{G}_b \implies |\hat{S}_{j_1,j_2}| \le \varepsilon,$$

ensuring no direct edges with strength greater than $\epsilon$ can exist across groups.

**B. Empirical third-order moment tensor**

**for** $\ell = 1, 2, 3$ **do**
  $X_\ell \leftarrow$ columns of $J$ indexed by $\mathcal{G}_\ell$ ▷ $X_\ell \in \mathbb{R}^{n\times|\mathcal{G}_\ell|}$
**end for**
Compute

$$\hat{T} = \frac{1}{n}\sum_{i=1}^{n} X_1^{(i)} \otimes X_2^{(i)} \otimes X_3^{(i)} \ \in\ \mathbb{R}^{|\mathcal{G}_1|\times|\mathcal{G}_2|\times|\mathcal{G}_3|}.$$

**C. Tensor decomposition**

Run a CP tensor-power decomposition on $\hat{T}$ to obtain $k$ = 4 components $\{(\hat{\pi}_{qc}, \hat{\mu}_{qc}^{(1)}, \hat{\mu}_{qc}^{(2)}, \hat{\mu}_{qc}^{(3)})\}_{q,c\in\{0,1\}^2}$, where $\hat{\pi}_{qc} > 0$ and $\hat{\mu}_{qc}^{(\ell)} \in \mathbb{R}^{|\mathcal{G}_\ell|}$.

**D. Assemble full means**

**for** $q, c \in \{0,1\}^2$ **do**
  $\hat{\mu}_{qc} \leftarrow \mathrm{concat}\big(\hat{\mu}_{qc}^{(1)}, \hat{\mu}_{qc}^{(2)}, \hat{\mu}_{qc}^{(3)}\big) \in \mathbb{R}^p$.
**end for**

**E. State alignment with anchors**

Find the permutation $\rho$ of $\{1,\dots,4\}$ that minimizes $\sum_{r=1}^4 \big\|\hat{\mu}_{\rho(r)} - \mu_{\text{anchor}(r)}\big\|_2^2$, where the four anchor prototypes correspond to $(Q, C) = \{00, 01, 10, 11\}$.

$(\hat{\mu}_{00}, \hat{\mu}_{01}, \hat{\mu}_{10}, \hat{\mu}_{11}) \leftarrow (\hat{\mu}_{\rho(1)}, \hat{\mu}_{\rho(2)}, \hat{\mu}_{\rho(3)}, \hat{\mu}_{\rho(4)})$.

**F. Mixing weights**

$(\hat{\pi}_{00}, \hat{\pi}_{01}, \hat{\pi}_{10}, \hat{\pi}_{11}) \leftarrow (\hat{\pi}_{\rho(1)}, \hat{\pi}_{\rho(2)}, \hat{\pi}_{\rho(3)}, \hat{\pi}_{\rho(4)})$.

**return** $\{\hat{\mu}_{qc}, \hat{\pi}_{qc}\}_{q,c\in\{0,1\}}$.

then set $\hat{\mu}_{2,q,c} = (V_{12}\,\Sigma_{12}^{1/2})\,R^\top$ and $\hat{\mu}_{3,q,c} = (V_{13}\,\Sigma_{13}^{1/2})\,R^\top$ to align all three views.

This SVD baseline recovers $\{\mu_{v,q,c}\}$ up to the permutation/sign ambiguity inherent in any orthogonal transform.

### C.5 Genral CARE Setup

**Extension Beyond the Gaussian Observation Model.** The multivariate-Gaussian assumption for $J|H$ is convenient—its first two or three moments already encode all information needed for the sparse + low-rank and tensor steps—but it is not a requirement. Because CARE learns the *graphical* structure, the same pipeline applies whenever each judge's conditional distribution lies in an exponential family or, more generally, a latent-variable generalized linear model (GLM):

- **Categorical or ordinal scores.** For Likert ratings or pairwise preferences we can set

$$J_i \mid H \ \sim\ \mathrm{Categorical}\big(\mathrm{softmax}(W_i^\top H)\big) \quad \text{or} \quad \mathrm{Ordinal-logit}(W_i^\top H).$$

  The graph—hence the sparse mask $S$—is unchanged; only the node-wise likelihoods differ. We still recover $S$ from conditional-mutual-information or pseudo-likelihood scores, and we still factorize higher-order indicator moments such as $\mathbb{E}\big[\mathbf{1}_{\{J_a=\ell\}}\ \mathbf{1}_{\{J_b=m\}}\ \mathbf{1}_{\{J_c=n\}}\big]$.

- **Mixed Discrete-Continous Scores.** When some judges output real scores and others categorical flags, we use a mixed conditional distribution:

$$p(J|H) = \left[\Pi_{i\in\text{Cont.}}\mathcal{N}(J_i;\mu_{H_i},\sigma_i^2)\right]\left[\Pi_{j\in\text{Disc.}}\text{Bernoulli}(\sigma(W_j^\top H))\right].$$

  CARE forms mixed raw/indicator moments, and identifiability again follows from standard tensor-decomposition guarantees for mixed conditional means.
- **Heavy-tailed or skewed real scores.** When numeric scores are skewed or contain outliers, a multivariate-$t$ or Gaussian scale mixture is appropriate. Up to a scalar factor, the covariance still decomposes as sparse + low-rank, so Steps 1–2 of Algorithm 2 work after a simple rescaling.

Empirically, we find that replacing the Gaussian local likelihood only affects the estimation of sparse structure and extraction of latent factors, not the subsequent symmetry-breaking or posterior computation; thus the overall CARE pipeline generalizes with minimal adjustments.

### C.6 Heuristics and Justifications

**Heuristics for symmetry breaking** Any instantiation of CARE will require symmetry-breaking procedures for latent variable identifiability. For example, the fully Gaussian case needs a heuristic to identify the true-quality direction among latent factors, distinguishing $Q$ from confounders $C$. In the binary-Gaussian mixture scenario, an additional step resolves ambiguity between latent states ($Q = 0$ vs. $Q = 1$). Doing so will require additional information that can come from modeling assumptions, the use of ground-truth samples, or heuristics. We detail some examples below:

- **Identifying True-Quality Factor for Joint-Gaussian Model.** We introduce heuristics particularly aimed at distinguishing the true-quality latent variable from confounding latent variables. First, the *human-anchor criterion* leverages a small validation set containing human ratings. By including these human judgments in the graphical model, we anchor the latent quality variable to ground truth by selecting the latent factor exhibiting the strongest connection to the human evaluations. Second, we apply a *loading balance heuristic*, identifying the true-quality factor as one that loads broadly and with similar magnitude across all competent judges. Conversely, factors dominated by a few judges typically indicate shared confounding rather than true quality.
- **Identifying Latent States for Mixed Model.** In scenarios such as the tensor-based method, symmetry breaking additionally involves distinguishing latent states corresponding to different quality levels (e.g., $Q = 0$ versus $Q = 1$). In practice, we can use known labeled samples (such as high-quality examples) to anchor and identify latent-state configurations. By comparing different latent configurations with these known labeled samples, we select the latent-state assignment that best aligns with empirical observations, effectively removing latent state ambiguity.

**Heuristic for Addressing Orthogonality Violations in CARE (SVD).**

Existing heuristics for identifying the true quality latent factor can estimate corresponding weights, but they often suffer from bias when the orthogonality assumption—central to the application of SVD—is violated. This issue commonly arises in real-world datasets. We found the following weighting rule effective in both synthetic and real-world settings:

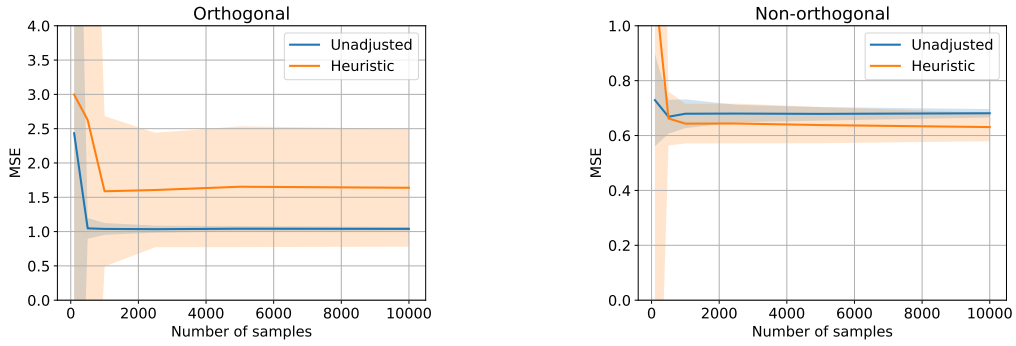
$$w \leftarrow \lambda^\star u^\star - \sum_{u_i\in U\setminus\{u^\star\}} \lambda_i u_i,$$

where $w$ represents the learned weights for each judge, $\lambda^*$ and $u^*$ is the singular value and vector of $L$ that corresponds to the direction that is closest to true quality latent variable, $\lambda_i, u_i$ represent rest of the singular values and vectors, which can be interpreted as spurious/confounding factors.

This rule intuitively subtracts the influence of overlapping (non-orthogonal) confounding components from the estimated true score factor.

Figure 4 illustrates the effect of this heuristic in a synthetic fully Gaussian setup. In the non-orthogonal case—where confounding components overlap with the true signal—the heuristic improves the estimation of the true latent variable. In contrast, it underperforms in the orthogonal case, where judges influenced by true scores are cleanly separated from those influenced by confounders.

**Justification of Decomposing Covariance Matrix.** In the joint-Gaussian setting we decompose the *precision* matrix, whose sparsity pattern directly encodes conditional independences in an undirected




(a) Orthogonal

(b) Non-Orthogonal

Figure 4: Effect of the proposed heuristic in a fully Gaussian synthetic setup. We estimate the true quality variable $Q$ and report the mean squared error. The heuristic improves estimation in the non-orthogonal setting, but slightly degrades performance in the orthogonal setting where true and confounding components are disjoint.

graphical model. For a *mixed* Gaussian model, however, each observation $J \in \mathbb{R}^p$ is generated by first drawing a latent class label $Q, C \in \{0, 1\}^2$ (with probabilities $\pi_{qc}$) and then sampling

$$J \mid Q, C = q, c \;\sim\; \mathcal{N}\big(\mu_{qc},\, \Sigma\big),$$

where the within-component covariance $\Sigma$ does not depend on $q, c$. Because the latent variable only perturbs the mean, the marginal covariance of $J$ splits, via the *law of total covariance*, into

$$\mathrm{Cov}(J) \;=\; \underbrace{\mathbb{E}\big[\mathrm{Cov}(J \mid Q, C)\big]}_{=\Sigma} \;+\; \underbrace{\mathrm{Cov}\big(\mathbb{E}[J \mid Q, C]\big)}_{=\sum_{q,c} \pi_{qc}\,(\mu_{qc}-\bar{\mu})(\mu_{qc}-\bar{\mu})^\top}, \quad \bar{\mu} := \sum_{q,c} \pi_{qc}\mu_{qc}. \tag{2}$$

The first term, $\Sigma$, is the same sparse block-diagonal matrix we plant in the simulator to model direct judge–judge interactions; the second term is an outer-product mixture of at most $4$ linearly independent directions and hence has rank $\leq 4$. Equation 2 therefore exhibits the population covariance as a *sparse* + *low-rank* decomposition,

$$\mathrm{Cov}(J) \;=\; S \;+\; L, \quad S = \Sigma \text{ (sparse)}, \quad L = \mathrm{Cov}\big(\mathbb{E}[J \mid Q, C]\big) \text{ (low rank)}.$$

Because sparsity now lives in $S$, not in the inverse covariance, estimating $S$ and $L$ by fitting a sparse-plus-low-rank model directly to the empirical covariance is both natural and statistically identifiable for the mixed Gaussian case.

# D Theory

We formalize the graphical model under joint gaussian distribution and notation (Section D.1), then discuss the identifiability of graph structure with exact and approximate recovery (Section D.2) and quantify the sample complexity required for consistent recovery of our SVD-based algorithm (Section D.3). Next, we present the model misspecification error when confounding factor is not correctly characterized (Section D.4). Finally, we discuss sample complexity required for tensor-based algorithm under mixed Gaussian distribution (Section D.5). All proofs are included in Section D.6.

## D.1 Model and Notation

We discuss the model under joint-gaussian distribution where all variables follow the same definitions as in Section 2. Briefly, $J = (J_1, \ldots, J_p)^\top$ stacks the $p$ observable judge scores, and $H = (Q, C_1, \ldots, C_k)^\top$ collects the $h = k + 1$ latent variables.

$$\Sigma = \mathrm{Cov}\big[(J, H)^\top\big], \qquad \Sigma^{-1} = K = \begin{pmatrix} K_{JJ} & K_{JH} \\ K_{HJ} & K_{HH} \end{pmatrix},$$

where the subscript $J$ (resp. $H$) refers to observable (resp. latent) coordinates.

The observable block factorizes via the Schur complement:

$$(\Sigma_{JJ})^{-1} = S + L, \quad S = K_{JJ}, \quad L = K_{JH}\, K_{HH}^{-1}\, K_{HJ}.$$

Here $\Sigma_o$ is the covariance matrix of observable variables, $S \in \mathbb{R}^{p\times p}$ is sparse and encodes direct conditional edges among judges, $L$ is low-rank with $rank(L) \leq h$ and captures dependencies mediated by the latent variables. Entry $(K_{JH})_{i\ell}$ is the edge weight between judge $i$ and latent factor $\ell$.

## D.2 Graph Structure Identifiability

While $(S, L)$ can be recovered (e.g. via convex sparse-plus-low-rank regularization [36], the finer structure of $K_{JH}$ is usually not identifiable from $L$. For example, for arbitrary rotation matrix $R \in \mathbb{R}^{h\times h}$, $L = (K_{JH}K_{HH}^{-1/2}R)(R^\top K_{HH}^{-1/2}K_{HJ})$, this indicates one cannot distinguish $K_{JH}K_{HH}^{-1/2}$ from $K_{JH}K_{HH}^{-1/2}R$ without further constraints. Hence, we need to impose additional assumptions:

**Assumption D.1** (Latent–latent independence and eigen-gap)**.** $K_{HH} = \mathrm{diag}(d_1, \ldots, d_h)$ with $d_1 > d_2 > \cdots > d_h > 0$.

**Assumption D.2** (Orthogonal latent–observable connections)**.** The columns of $K_{JH}$ are orthogonal, i.e. $K_{JH}^\top K_{JH}$ is diagonal. A special case is the *disjoint-support* model where each judge connects to exactly one latent factor.

Next, we provide an exact recovery result given the above assumptions.

**Theorem D.3** (Exact Recovery)**.** *Under Assumptions 1 and 2, columns in $K_{JH}$ are identifiable up to column permutations and sign flips.*

Real-world data rarely satisfy the exact orthogonality in Assumption D.2. To assess robustness, consider the following perturbed connection matrix:

$$\tilde{K}_{JH} = K_{JH} + E, \qquad \|E\|_2 \text{ small}.$$

The associated low-rank part is $\tilde{L} = \tilde{K}_{JH}K_{HH}^{-1}\tilde{K}_{HJ}$. Let the eigen-pairs of $L = K_{JH}K_{HH}^{-1}K_{HJ}$ and $\tilde{L}$ be $\{(\lambda_i, u_i)\}_{i=1}^h$ and $\{(\tilde{\lambda}_i, \tilde{u}_i)\}_{i=1}^h$, ordered so that $\lambda_1 > \cdots > \lambda_h > 0$, and denote the eigen-gap by

$$\delta_i = \min_{j\neq i} |\lambda_i - \lambda_j| > 0.$$

**Theorem D.4** (Stability under approximate orthogonality)**.** *For every $i \in [h]$,*

$$\|\hat{u}_i - u_i\|_2 \;\leq\; \frac{2\|K_{HH}^{-1}\|_2\, \|E\|_2}{\delta_i} \;+\; O\big(\|E\|_2^2\big).$$

This indicates that latent–observable directions remain identifiable (up to column permutations and sign flips) whenever the perturbation norm $\|E\|_2$ is small relative to the eigen-gap $\delta_i$. We defer the proof to Appendix D.6.

### D.3 Sample Complexity Bound

We now quantify how many i.i.d. samples are needed for the two–stage estimator in Algorithm 2 to recover the latent–observable directions $K_{JH} \in \mathbb{R}^{p \times h}$.

As detailed in Algorithm 2, our estimator for $K_{JH}$ proceeds in two stages: first, a sparse + low-rank decomposition of sample precision matrix. Second, we extract the latent–observable directions by taking the rank-$h$ eigen-decomposition $\hat{L}_n = \sum_{i=1}^{h} \hat{\lambda}_i \, \hat{u}_i \hat{u}_i^\top$ and setting $\hat{K}_{JH} := [\hat{u}_1, \ldots, \hat{u}_h]$.

**Theorem D.5** (Sample complexity for recovering $K_{JH}$)**.** *Let $L^* = K_{JH} K_{HH}^{-1} K_{HJ} \in \mathbb{R}^{p \times p}$ have distinct eigenvalues $\lambda_1 > \cdots > \lambda_h$ and define the (global) eigengap $\delta := \min_{1 \le i < j \le h} |\lambda_i - \lambda_j|$. Assume the identifiability, incoherence, and curvature conditions of [36]. Then for any $\epsilon > 0$, with probability at least $1 - 2e^{-\epsilon}$,*

$$\max_{i \le h} \left\| \hat{u}_i - u_i \right\|_2 \;=\; O\Big( \frac{\sqrt{\epsilon}}{\sqrt{n}\, \xi(T)\, \delta} \Big),$$

*where $n$ is the sample size, $\hat{u}_i$ and $u_i$ are the $i$-th eigenvectors of $\hat{L}_n$ and $L^*$ respectively. $T = T(L^*)$ is the tangent space of $L^*$, $\xi(T)$ is the curvature constant from [36].*

We defer the proof to Appendix D.6. At a high-level, we adapt the identifiability, incoherence and curvature conditions from Theorem 4.1 of [36] and combine it with extended result of Davis-Khan's theorem [38].

This bound shows that the column-wise $\ell_2$ error decays at the standard parametric rate $n^{-1/2}$, and is attenuated by both the manifold curvature $\xi(T)$ and the eigengap $\delta$. Achieving an accuracy of at most $\alpha \in (0, 1)$ therefore requires

$$n = \tilde{O}\big( \frac{\epsilon}{\xi(T)^2 \delta^2 \alpha^2} \big)$$

samples, up to universal constants and log-factors.

### D.4 Misspecification Error

Many label aggregation frameworks (e.g.,[39, 28, 29]) assume a *single* latent variable that explains the observed labels. However, in setups like LLM-as-a-judge, the scores may be influenced by additional latent factors or confounders that also affect the observed annotations. Ignoring these *confounder* latents leads to model misspecification, which can bias the aggregated labels. We characterize this bias and analyze its impact on the estimated aggregation weights.

Let $L^* = \sum_{\ell=1}^{h} \frac{1}{d_\ell} \mathbf{k}_\ell \mathbf{k}_\ell^T$ be the true rank-$h$ low-rank component of the observable precision matrix, derived from the latent-observable connection matrix $K_{JH} = [\mathbf{k}_1, \ldots, \mathbf{k}_h]$ and latent-latent precision $K_{HH} = \text{diag}(d_1, \ldots, d_h)$. Let $\mathbf{u}_1^{\text{true}} = \mathbf{k}_1 / ||\mathbf{k}_1||_2$ be the true direction of influence for the quality score latent variable $Q$ (assuming $\mathbf{k}_1 \neq \mathbf{0}$).

Define $\mathbf{A} = \frac{1}{d_1} \mathbf{k}_1 \mathbf{k}_1^T$. Its principal (and only non-zero) eigenvalue is $\lambda_1 = \frac{1}{d_1} ||\mathbf{k}_1||_2^2$, and its spectral gap (to its other zero eigenvalues) is $\delta = \lambda_1$. Let $\mathbf{E} = \sum_{\ell=2}^{h} \frac{1}{d_\ell} \mathbf{k}_\ell \mathbf{k}_\ell^T$ be the confounding component, so $L^* = \mathbf{A} + \mathbf{E}$. Let $\mathbf{v}_1$ be the principal unit-norm eigenvector of $L^*$. When a rank-1 model is fitted, the estimated direction is $\hat{\mathbf{u}}_1^{\text{pop}} = \mathbf{v}_1$.

**Theorem D.6.** *If $||\mathbf{E}||_{op} \le \delta/2$, the $\ell_2$ deviation of the estimated direction $\mathbf{v}_1$ from $\mathbf{u}_1^{\text{true}}$ is bounded by:*

$$\left|\left| \mathbf{v}_1 - s \mathbf{u}_1^{\text{true}} \right|\right|_2 \le \frac{2\, ||\mathbf{E}||_{op}}{\delta} = \frac{2 \left|\left| \sum_{\ell=2}^{h} \frac{1}{d_\ell} \mathbf{k}_\ell \mathbf{k}_\ell^T \right|\right|_{op}}{\frac{1}{d_1} \, ||\mathbf{k}_1||_2^2}$$

*for a sign $s = \pm 1$ (chosen so that $s(\mathbf{u}_1^{\text{true}})^T \mathbf{v}_1 \ge 0$).*

*Proof.* By Davis-Kahan theorem (Theorem 2 in [38]), if $||\mathbf{E}||_{\text{op}} \le \delta/2$, then the $\ell_2$ distance between $\mathbf{v}_1$ and $\mathbf{u}_1^{\text{true}}$ (after aligning their signs via $s = \pm 1$) is bounded by:

$$\left|\left| \mathbf{v}_1 - s \cdot \mathbf{u}_1^{\text{true}} \right|\right|_2 \le \frac{2\, ||\mathbf{E}||_{\text{op}}}{\delta}.$$

Plugging in $E$ yields the desired result:

$$\left|\left|\mathbf{v}_1 - s \cdot \mathbf{u}_1^{\text{true}}\right|\right|_2 \leq \frac{2\left|\left|\sum_{\ell=2}^{h} \frac{1}{d_\ell}\mathbf{k}_\ell\mathbf{k}_\ell^T\right|\right|_{\text{op}}}{\frac{1}{d_1}\left|\left|\mathbf{k}_1\right|\right|_2^2}.$$

□

The theorem quantifies the directional bias in the estimated influence of $Q$ when confounders are ignored. This bias is proportional to the collective "strength" of confounders in the precision domain (numerator) and inversely proportional to $Q$'s own "strength" (denominator). Fitting a rank-1 model forces this bias, while a higher-rank model offers the capacity to separate these influences.

**Corollary D.7** (Error Bound for Estimated Conditional Mean of $Q$)**.** *Denote the true conditional mean of true quality score latent variable $Q$ given the observable variables $O = (J_1, ..., J_p)$ be denoted by $\mathbb{E}[Q|O]_{\text{true}}$. Then, $\mathbb{E}[Q|\mathbf{o}]_{\text{true}} = -\frac{||\mathbf{k}_1||_2}{d_1}(\mathbf{u}_1^{\text{true}})^T\mathbf{o}$. Let an estimated conditional mean with the misspecified direction, $\mathbb{E}[Q|\mathbf{o}]_{\text{mis}}$, be formed using the misspecified direction $\mathbf{v}_1$ be $\mathbb{E}[Q|\mathbf{o}]_{\text{mis}} = -\frac{||\mathbf{k}_1||_2}{d_1}(s \cdot \mathbf{v}_1)^T\mathbf{o}$, where $s = \pm 1$ is chosen such that $s \cdot (\mathbf{u}_1^{\text{true}})^T\mathbf{v}_1 \geq 0$. Then, the absolute error in the estimated conditional mean due to the directional misspecification is bounded by:*

$$|\mathbb{E}[Q|\mathbf{o}]_{\text{mis}} - \mathbb{E}[Q|\mathbf{o}]_{\text{true}}| \leq \frac{2\left|\left|\sum_{\ell=2}^{h} \frac{1}{d_\ell}\mathbf{k}_\ell\mathbf{k}_\ell^T\right|\right|_{op}}{\left|\left|\mathbf{k}_1\right|\right|_2}\left|\left|\mathbf{o}\right|\right|_2$$

*This holds if the condition from the main theorem, $\left|\left|\mathbf{E}\right|\right|_{op} \leq \delta/2 = \frac{1}{2d_1}\left|\left|\mathbf{k}_1\right|\right|_2^2$, is met, where $\mathbf{E} = \sum_{\ell=2}^{h} \frac{1}{d_\ell}\mathbf{k}_\ell\mathbf{k}_\ell^T$.*

*Proof.* The absolute difference is:

$$\begin{aligned}
|\mathbb{E}[Q|\mathbf{o}]_{\text{mis}} - \mathbb{E}[Q|\mathbf{o}]_{\text{true}}| &= \left|-\frac{||\mathbf{k}_1||_2}{d_1}(s \cdot \mathbf{v}_1)^T\mathbf{o} - \left(-\frac{||\mathbf{k}_1||_2}{d_1}(\mathbf{u}_1^{\text{true}})^T\mathbf{o}\right)\right| \\
&= \left|-\frac{||\mathbf{k}_1||_2}{d_1}(s \cdot \mathbf{v}_1 - \mathbf{u}_1^{\text{true}})^T\mathbf{o}\right| \\
&= \frac{||\mathbf{k}_1||_2}{d_1}\left|(s \cdot \mathbf{v}_1 - \mathbf{u}_1^{\text{true}})^T\mathbf{o}\right|
\end{aligned}$$

By the Cauchy-Schwarz inequality, $\left|(\mathbf{x})^T\mathbf{y}\right| \leq ||\mathbf{x}||_2 \, ||\mathbf{y}||_2$. Applying this:

$$|\mathbb{E}[Q|\mathbf{o}]_{\text{mis}} - \mathbb{E}[Q|\mathbf{o}]_{\text{true}}| \leq \frac{||\mathbf{k}_1||_2}{d_1}\left|\left|s \cdot \mathbf{v}_1 - \mathbf{u}_1^{\text{true}}\right|\right|_2 ||\mathbf{o}||_2$$

The term $||s \cdot \mathbf{v}_1 - \mathbf{u}_1^{\text{true}}||_2$ is equivalent to $||\mathbf{v}_1 - s \cdot \mathbf{u}_1^{\text{true}}||_2$ from the main theorem statement, where $s$ aligns $\mathbf{u}_1^{\text{true}}$ with $\mathbf{v}_1$. From the preceding Theorem, we have the bound (where $\delta = \frac{1}{d_1}||\mathbf{k}_1||_2^2$):

$$\left|\left|\mathbf{v}_1 - s \cdot \mathbf{u}_1^{\text{true}}\right|\right|_2 \leq \frac{2\,||\mathbf{E}||_{\text{op}}}{\delta} = \frac{2\left|\left|\sum_{\ell=2}^{h} \frac{1}{d_\ell}\mathbf{k}_\ell\mathbf{k}_\ell^T\right|\right|_{\text{op}}}{\frac{1}{d_1}||\mathbf{k}_1||_2^2}$$

Substituting this bound into the inequality for the error in the conditional mean:

$$\begin{aligned}
|\mathbb{E}[Q|\mathbf{o}]_{\text{mis}} - \mathbb{E}[Q|\mathbf{o}]_{\text{true}}| &\leq \frac{||\mathbf{k}_1||_2}{d_1}\left(\frac{2\,||\mathbf{E}||_{\text{op}}}{\frac{1}{d_1}||\mathbf{k}_1||_2^2}\right)||\mathbf{o}||_2 \\
&= \frac{||\mathbf{k}_1||_2}{d_1} \cdot \frac{2d_1\,||\mathbf{E}||_{\text{op}}}{||\mathbf{k}_1||_2^2} \cdot ||\mathbf{o}||_2 \\
&= \frac{2\,||\mathbf{E}||_{\text{op}}}{||\mathbf{k}_1||_2}||\mathbf{o}||_2 \\
&= \frac{2\left|\left|\sum_{\ell=2}^{h} \frac{1}{d_\ell}\mathbf{k}_\ell\mathbf{k}_\ell^T\right|\right|_{\text{op}}}{||\mathbf{k}_1||_2}||\mathbf{o}||_2
\end{aligned}$$

□

This corollary shows that the error in the estimated conditional mean of $Q$ (due to using the misspecified direction for $Q$'s influence) scales with:

- The magnitude of the observable vector $\mathbf{o}$ (specifically, $||\mathbf{o}||_2$).
- The collective strength of the confounding latent variables in the precision domain ($\left|\left|\sum_{\ell=2}^{h} \frac{1}{d_\ell}\mathbf{k}_\ell\mathbf{k}_\ell^T\right|\right|_{\text{op}}$).
- Inversely with the $\ell_2$-norm of the true connection weights of $Q$ ($||\mathbf{k}_1||_2$).

Especially, we see that strong confounders widen the gap bound, whereas heavier connection weights to the true score shrink it. Put differently, ***misspecification hurts most when confounders are strong and the quality signal is weak.***

### D.5 Sample Complexity for CARE tensor algorithm

**Assumption D.8** (Model and identifiability)**.** Let $J = (X_1^\top, X_2^\top, X_3^\top)^\top \in \mathbb{R}^p$ ($p = p_1 + p_2 + p_3$) be one observations i.i.d generated as

$$(Q, C) \sim \text{Multinomial}(\{\pi_{qc}\}_{q,c\in\{0,1\}}), \qquad X_\ell \mid (Q = q, C = c) \;\sim\; \mathcal{N}\big(\mu_{qc}^{(\ell)},\, \Sigma\big),$$

with $\ell \in \{1, 2, 3\}$. Write $r \in [4] \leftrightarrow (q, c) \in \{0, 1\}^2$ and define $w_r := \pi_{qc}$, $a_r := \mu_{qc}^{(1)} \in \mathbb{R}^{p_1}$, $b_r := \mu_{qc}^{(2)} \in \mathbb{R}^{p_2}$, $c_r := \mu_{qc}^{(3)} \in \mathbb{R}^{p_3}$.

(A1) **Block-conditional independence.** $X_1 \perp X_2 \perp X_3 \mid (Q, C)$.

(A2) **Full-rank moment tensor.** The population third-order moment $M := \mathbb{E}[X_1 \otimes X_2 \otimes X_3] = \sum_{r=1}^{4} w_r\, a_r \otimes b_r \otimes c_r$ has rank 4, with $\pi_{\min} := \min_r \pi_r > 0$ and $\lambda_{\min} := \min_r \|a_r\|_2\|b_r\|_2\|c_r\|_2 > 0$.

(A3) **Non-degenerate covariance.** $\sigma_{\max}^2 := \|\Sigma\|_{\text{op}} < \infty$.

(A4) **Spectral gap.** The CP factors are uniquely defined up to scaling/sign and satisfy the eigenvalue-gap condition of Theorem 5.1 in [37]. Denote that gap by $\delta > 0$.

(A5) **Correct graph partition.** There exist a graph partition such that judges between different groups are conditional independent. Step A of Algorithm 3 returns the true groups $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$.

**Theorem D.9** (Sample complexity of CARE tensor step)**.** *Fix $0 < \varepsilon < 1$ and let the assumptions above hold. Run Algorithm 2 (CARE) on $n$ i.i.d. samples to obtain $\{\hat{\mu}_{qc}, \hat{\pi}_{qc}\}_{q,c\in\{0,1\}}$. Under Assumption D.8, there exist universal constants $C_1, C_2 > 0$ such that if*

$$n \;\geq\; C_1\, \frac{\sigma_{\max}^6}{\delta^2\, \pi_{\min}^2}\; p\, \log\!\big(p/\varepsilon\big),$$

*then with probability at least $1 - \varepsilon$*

$$\max_{q,c} \big\|\hat{\mu}_{qc} - \mu_{qc}\big\|_2 \;\leq\; C_1\, \frac{\sigma_{\max}^3}{\delta}\, \sqrt{\frac{p\, \log(p/\varepsilon)}{n}}, \qquad \max_{q,c} \big|\hat{\pi}_{qc} - \pi_{qc}\big| \;\leq\; C_2 \sqrt{\frac{p\, \log(p/\varepsilon)}{n}}.$$

We defer the proof to D.6.

### D.6 Proofs

#### Proof of Theorem D.3

*Proof.* Let low-rank matrix satisfies $L = \sum_{i=1}^{h} d_i\, u_i u_i^\top$ with $u_i$ the $i$-th column of $K_{oh}$. By Assumption D.2 the $u_i$ are mutually orthogonal, and by Assumption D.1 the eigenvalues $d_1 > \cdots > d_h$ are distinct; hence this rank-1 decomposition is the (unique) spectral decomposition of $L$. Thus each $u_i$ is identifiable from $L$ up to sign and ordering, proving the theorem. □

**Proof of Theorem D.4**

*Proof.* We apply standard matrix perturbation theory for eigenvectors. Starting from the eigenvalue decomposition:

$$L\, u_i = \lambda_i\, u_i,$$

we write the perturbed matrix as

$$\tilde{L} = (K_{oh} + E)K_{hh}^{-1}(K_{oh} + E)^\top = L \;+\; K_{oh}K_{hh}^{-1}E^\top \;+\; EK_{hh}^{-1}K_{oh}^\top \;+\; EK_{hh}^{-1}E^\top.$$

Let $\Delta L = \tilde{L} - L$. By the Davis–Kahan theorem,

$$\|\hat{u}_i - u_i\|_2 \;\le\; \frac{2\,\|\Delta L\|_2}{\delta_i},$$

where $\delta_i = \min_{j\neq i} |\lambda_i - \lambda_j| > 0$. Moreover,

$$\|\Delta L\|_2 \;\le\; 2\,\|K_{oh}\|_2\,\|K_{hh}^{-1}\|_2\,\|E\|_2 \;+\; O(\|E\|_2^2)$$

and $\|K_{oh}\|_2 = 1$. Hence

$$\|\hat{u}_i - u_i\|_2 \;\le\; \frac{2\,\|K_{hh}^{-1}\|_2\,\|E\|_2}{\delta_i} \;+\; O(\|E\|_2^2).$$

This completes the proof. □

**Proof of Theorem D.5**

*Proof of Theorem D.5.* **Step 1 – Spectral error of $\hat{L}_n$.** Apply Chandrasekaran et al.'s Theorem 4.1 with the regularization parameters

$$\gamma_n \;=\; \frac{48\sqrt{2}D\psi(2-\nu)}{\xi(T)\nu}\sqrt{\frac{\epsilon}{n}}, \qquad \sigma, \theta \text{ as in their conditions (3)–(4).}$$

Under the incoherence and curvature conditions of their Proposition 3.3, there exists a universal constant $C_1 > 0$ such that, with probability at least $1 - 2e^{-\epsilon}$,

$$\left\|\, \hat{L}_n - L^* \right\|_2 \;\le\; C_1\, \frac{\sqrt{\epsilon/n}}{\xi(T)}. \tag{3}$$

**Step 2 – Eigenvector perturbation via Davis–Kahan.** Let $L^* = U\Lambda U^\top$ with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_h, 0, \ldots, 0)$ and collect the top–$h$ eigenvectors in $U_h = [u_1, \ldots, u_h]$. Write the spectral decomposition of the estimator as $\hat{L}_n = \hat{U}_h\hat{\Lambda}\hat{U}_h^\top + R$, where $R$ contains only the eigen-components of rank $> h$. Set the perturbation $E := \hat{L}_n - L^*$.

Applying Corollary 3 from [38] to the $i$-th eigenpair gives

$$\|u_i - \hat{u}_i\|_2 \;\le\; \frac{2^{3/2}\|E\|_2}{\delta_i}. \tag{4}$$

**Step 3 – Combine the two bounds.** Insert equation 3 into equation 4:

$$\|\,\hat{u}_i - u_i\,\|_2 \;\le\; \frac{2^{3/2}C_1}{\delta\,\xi(T)}\sqrt{\frac{\epsilon}{n}} \qquad \forall\, i \in [h],$$

and take the maximum over $i$. This proves the advertised high-probability bound

$$\max_{i\le h} \|\,\hat{u}_i - u_i\,\|_2 \;=\; O\!\Big(\tfrac{\sqrt{\epsilon/n}}{\xi(T)\,\delta}\Big).$$

**Step 4 – Invert to a sample-size requirement.** Setting the right-hand side to a target accuracy $\varepsilon \in (0,1)$ and solving for $n$ yields $n \;\ge\; \frac{4C_1^2}{\varepsilon^2}\,\frac{\epsilon}{\xi(T)^2\delta^2}$, which is the sample-complexity statement in the theorem. □

**Proof for Theorem D.9**

*Proof sketch.* **Step 1: Concentration of the empirical tensor.** Let $\hat{M} := \frac{1}{n}\sum_{i=1}^{n} X_1^{(i)} \otimes X_2^{(i)} \otimes X_3^{(i)}$. Because each $X_\ell$ is sub-Gaussian with proxy $\sigma_{\max}$, the operator-norm Bernstein bound for order-3 tensors (Lemma 5 of 40) yields

$$\|\hat{M} - M\|_{\mathrm{op}} \;=\; O\Big(\sigma_{\max}^3 \sqrt{\tfrac{p\,\log(p/\varepsilon)}{n}}\Big) \quad \text{w.p. } 1-\varepsilon/2.$$

**Step 2: Robust CP decomposition.** Applying the non-symmetric tensor power method of [37, Alg. 2] to $\hat{M}$ and invoking their perturbation bound (Theorem 5.1 therein) gives, for every component $r \in [4]$,

$$\big\|(\hat{a}_r, \hat{b}_r, \hat{c}_r) - (a_r, b_r, c_r)\big\|_2 \;=\; O\Big(\tfrac{1}{\delta}\,\|\hat{M} - M\|_{\mathrm{op}}\Big).$$

**Step 3: Assembling full means.** Algorithm 3 concatenates the three block-means, so $\hat{\mu}_r - \mu_r = (\hat{a}_r - a_r, \hat{b}_r - b_r, \hat{c}_r - c_r)$, and the same $O(\cdot)$ factor carries through.

**Step 4: Mixing-weight estimation.** Given accurate factor recovery, the usual least-squares re-estimation of weights satisfies $|\hat{\pi}_{qc} - \pi_{qc}| = O\big(\|\hat{M} - M\|_{\mathrm{op}}\big)$ (37, Theorem B.1), yielding the stated rate.

**Step 5: Union bound.** Combine Steps 1–4 and union-bound over the four components to obtain the final probability $1 - \varepsilon$. □

# E Experimental Details and Extended Results

This appendix provides additional details on our experimental setup and supplementary analyses. We first describe the datasets, metrics, baseline methods, and the pool of LLM judges used in our evaluation (E.1), followed by the exact prompt templates used to elicit judgments (E.2). We then report the performance of individual LLM judges (E.3) and introduce our suite of programmatic judges, including their construction and standalone accuracy (E.4). Next, we examine ablations on prompt-based interventions (E.5) and evaluate robustness to confounding factors, covering both injected stylistic and semantic biases as well as controlled experiments with dummy judges (E.6–E.7). We then present additional validations: synthetic experiments demonstrating the benefits of tensor decomposition (E.8), graph-aware tensor decomposition (E.9), and a real-world Gaussian mixture setting on CivilComments (E.10). Together, these results expand on the main text and demonstrate the generality and robustness of our CARE framework.

## E.1 Setup.

**Datasets & Metrics.** We use FeedbackQA [41], UltraFeedback [42], and HelpSteer2 [43] datasets for response scoring. Performance is benchmarked using Mean Absolute Error (MAE) to measure numerical accuracy and Kendall's $\tau$ rank correlation [44] to evaluate ranking consistency, accommodating variations in judge scales and calibration.

- **FeedbackQA** [41]: question–answer pairs rated for helpfulness on a 1–5 scale. We use the validation set, treating the average of two human ratings as ground truth.
- **HelpSteer2** [43]: large-scale assistant responses annotated on multiple axes (0–4). We use the validation set and take the helpfulness score as ground truth.
- **UltraFeedback** [42]: responses rated 0–10 for overall quality, with scores aggregated from GPT-4 and human raters. We randomly sample 5,000 examples for evaluation.
- **Synthetic Dataset (Section E.8).** We construct a synthetic dataset with latent state probabilities set to $\pi_{qc} = [0.2, 0.2, 0.3, 0.3]$, corresponding to latent states $(Q, C)$ as $(0, 0), (0, 1), (1, 0), (1, 1)$ respectively. The judges are organized into three distinct groups, each containing four judges whose conditional means $\mu_{qc}$ are randomly drawn from a uniform distribution ranging between $1$ and $4$. Dependence structures are imposed explicitly: for judges independent of the true quality variable $Q$, we constrain their conditional means such that averages depend solely on the confounder $C$ (i.e., rows corresponding to $Q = 0$ and $Q = 1$ are identical for each $C$ state).

**Baselines.** We compare CARE to following baseline aggregation methods: (i) majority voting (MV), (ii) simple averaging (AVG) [11], (iii) discrete-based weak supervision (WS) [39], and (iv) continuous-based weak supervision (UWS) [29].

**LLM Judges.** We consider the following LLMs as judges to score responses: `Llama-3.2-1B` [45], `Llama-3.2-3B` [45], `Llama-3.1-8B-Instruct` [45], `Mistral-7B-Instruct-v0.3` [46], `Qwen3-0.6B` [47], `Qwen3-1.7B` [47], `Qwen3-4B` [47], `Qwen3-8B` [47], `Phi-4-mini-instruct` [48], `gemma-3-1b-it` [49], `gemma-3-4b-it` [49].

**Computing Resources.** We used a server equipped with an NVIDIA RTX 4090 (24GB). Generating LLM judge outputs took up to 3 hours per dataset. In contrast, the aggregation algorithms were efficient, completing in under 1 minute for datasets with approximately 5,000 rows.

## E.2 Prompt Templates

In this subsection we provide the prompts we used for collecting LLM judgements.

**LLM Judge Scoring Template (FeedbackQA, HelpSteer2, Ultrafeedback)**

You will be given a user_question and system_answer couple.

Your task is to provide a 'total rating' scoring how well the system_answer answers the user concerns expressed in the user_question. Give your answer as a float on a scale of {min_rating} to {max_rating}, where {min_rating} means that the system_answer is not

helpful at all, and {max_rating} means that the answer completely and helpfully addresses the question.

Provide your feedback as follows:

Feedback:::
Total rating: (your rating, as a float between {min_rating} and {max_rating})

Now here are the question and answer.

Question: {question}
Answer: {answer}

Feedback:::
Total rating:

**Intervened LLM Judge Scoring Template (Section 3.4)**

You are a careful and impartial evaluator. Your task is to rate how well the system_answer responds to the user_question, focusing on substance, correctness, and completeness.

Rate the answer on a scale from {min_rating} to {max_rating}, where:
- {min_rating} means the answer does not address the question or is misleading,
- {max_rating} means the answer fully and helpfully addresses the question with accurate, relevant, and concise information.

Avoid common judgment biases:
- Do **not** reward irrelevant fluency, length, or politeness.
- Do **not** penalize answers solely for brevity if they are complete.
- Do **not** be influenced by phrasing style unless it affects clarity or correctness.

Judge only the answer's content in relation to the question. Ignore formatting, grammar, or tone unless they impact understanding or utility.

Provide your feedback as follows:

Feedback:::
Total rating: (a float from {min_rating} to {max_rating})

Now, here are the inputs:

Question: {question}
Answer: {answer}

Feedback:::
Total rating:

**LLM Judge Toxicity Scoring Template (CivilComments)**

You will be given a user_question and system_answer couple.

Your task is to provide a 'total rating' scoring how well the system_answer answers the user concerns expressed in the user_question. Give your answer as a float on a scale of {min_rating} to {max_rating}, where {min_rating} means that the system_answer is not

helpful at all, and {max_rating} means that the answer completely and helpfully addresses the question.

Provide your feedback as follows:

Feedback:::
Total rating: (your rating, as a float between {min_rating} and {max_rating})

Now here are the question and answer.

Question: {question}
Answer: {answer}

Feedback:::
Total rating:

### E.3 Individual LLM Judge Performance

Table 5 shows the performance of individual LLM judges. Overall, we find that aggregation methods outperform or match the best single judge. Since the relative performance of individual LLM judges is not known a priori in practical settings, aggregation offers a useful approach for integrating judges in an agnostic manner.

Table 5: Performance of individual judges across datasets, as evaluated in Section 3.1.

| | **FeedbackQA** | | **HelpSteer2** | | **UltraFeedback** | |
|---|---|---|---|---|---|---|
| | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) |
| gemma-3-1b-it | 1.0073 | 0.2315 | 1.0666 | 0.0825 | 1.0606 | 0.1812 |
| gemma-3-4b-it | **0.7578** | 0.4537 | 0.9920 | 0.1402 | 0.8492 | 0.2309 |
| Llama-3.1-8B-Instruct | 0.8148 | 0.4341 | 1.1364 | 0.1261 | 0.8648 | 0.3194 |
| Llama-3.2-1B | 1.2219 | -0.0525 | 1.0049 | -0.0132 | 1.0119 | 0.0752 |
| Llama-3.2-3B | 1.0362 | 0.0051 | 0.9995 | 0.0251 | 1.1522 | 0.1648 |
| Mistral-7B-Instruct-v0.3 | 1.0244 | 0.4539 | 1.0793 | 0.1116 | 0.8572 | 0.1735 |
| Phi-4-mini-instruct | 0.8082 | **0.4557** | 1.0692 | 0.1576 | 0.8355 | 0.3147 |
| Qwen3-0.6B | 1.0969 | 0.2073 | 1.1255 | 0.0370 | 1.0233 | 0.1254 |
| Qwen3-1.7B | 1.1507 | 0.2485 | 1.0693 | 0.1049 | 1.1382 | 0.1926 |
| Qwen3-4B | 1.0999 | 0.2854 | **0.9675** | **0.2290** | **0.7088** | **0.3921** |
| Qwen3-8B | 1.0517 | 0.4417 | **0.9675** | 0.2094 | 0.7512 | 0.3140 |

### E.4 Programmatic Judges

Programmatic judges, synthesized by LLMs, distill and convert evaluation logic into interpretable, cheap-to-obtain program code [12, 50, 51]. These program judges provide specialized, independent assessments compared to using LLMs directly as evaluators. We integrate these judge sets into CARE to enhance evaluation signals.

We describe the creation of programmatic judges and the criteria they encode. Using OpenAI's GPT-4o [14], we generate judges with the following prompt:

**Programmatic Judge Template**

You are now a judge to evaluate LLM generated response with a given question. You will write your evaluation logic into code and generate python programs to return their scores. Higher represents better response quality. Consider complex criteria for assessing responses, leveraging third-party models, embedding models, or text score evaluators as needed.

Function signature: def _judging_function(self, question, response):

We synthesize 23 programs and select 10 representative ones for our experiments (see Section 3.2 and Section 3.3). These programs evaluate responses based on diverse criteria: (i) structure, (ii) readability, (iii) safety, (iv) relevance, and (v) factuality. For example:

- **Structure**: A judge counts transition markers (e.g., "therefore," "however") to assess coherence, with more markers indicating better quality.
- **Relevance**: A judge uses TF-IDF to convert questions and responses into vectors, computing cosine similarity to measure semantic alignment (see Program 1). Another employs semantic embeddings for similarity metrics (see Program 2).
- **Readability**: A judge leverages a third-party API to evaluate complexity, using metrics like the Flesch–Kincaid grade level (see Program 3).

All judging logic, conditions, and pre-defined keyword lists are generated by the LLM. Below, we provide examples to illustrate this approach.

```
def _lexical_overlap(self, question, response):
    """Compute lexical overlap using TF-IDF for relevance evaluation.
        """
```

```
    # Preprocess input question and response (e.g., lowercase, remove
        stopwords)
    question_clean = self._preprocess(question)
    response_clean = self._preprocess(response)

    # Return 0.0 if either input is empty after preprocessing
    if not question_clean.strip() or not response_clean.strip():
        return 0.0

    # Transform inputs to TF-IDF vectors using the vectorizer
    tfidf_matrix = self.tfidf_vectorizer.fit_transform([question_clean
        , response_clean])
    question_vec = tfidf_matrix[0]  # Extract question vector
    response_vec = tfidf_matrix[1]  # Extract response vector

    # Compute cosine similarity between vectors and return as float
    return float(cosine_similarity(question_vec, response_vec)[0][0])
```

Program 1: Lexical Overlap Computation using TF-IDF.

```
def _semantic_similarity_strong(self, question, response):
    """Compute semantic similarity between question and response."""
    # Return 0.0 if either input is empty
    if not question.strip() or not response.strip():
        return 0.0

    # Encode question and response into dense vectors using the
        embedding model
    question_embedding = self.semantic_embedding_strong_model.encode(
        question, max_length=4096
    )["dense_vecs"]
    response_embedding = self.semantic_embedding_strong_model.encode(
        response, max_length=4096
    )["dense_vecs"]

    # Compute dot product similarity between embeddings
    similarity = question_embedding @ response_embedding

    # Clamp similarity score between 0.0 and 1.0 and return as float
    return float(max(0.0, min(1.0, similarity)))
```

Program 2: Semantic Similarity using Embedding Model.

```
def _readability(self, response):
    """Calculate readability metrics for response."""
    # Compute readability scores using textstat library
    return {
        # Flesch Reading Ease (inverted: higher score means harder to
            read)
        "flesch_reading_ease": 100 - textstat.flesch_reading_ease(
            response),
        # SMOG Index (higher score indicates higher reading difficulty
            )
        "smog_index": textstat.smog_index(response),
    }
```

Program 3: Readability Metrics Calculation.

We report the performance of individual program judges in Table 6. While their standalone performance is limited, they provide useful signals for the integration strategies discussed in Sections 3.2 and 3.3.

Table 6: Performance of individual program judges across datasets, with (*) indicating judges selected in Section 3.2.

| | **FeedbackQA** | | **HelpSteer2** | | **UltraFeedback** | |
|---|---|---|---|---|---|---|
| | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) |
| factuality_check_score (*) | 1.9956 | 0.0872 | 1.1992 | 0.0075 | 1.1910 | 0.0492 |
| factuality_factKB_score (*) | 1.0343 | 0.2288 | 1.7180 | 0.0414 | 1.4342 | 0.1051 |
| readability_flesch_reading (*) | 1.2185 | 0.0431 | 2.5682 | 0.0445 | 2.5145 | 0.1396 |
| readability_smog (*) | 0.9805 | 0.1277 | 2.3286 | 0.0283 | 2.3122 | 0.1604 |
| relevance_bleu | 1.4035 | 0.0126 | 2.7452 | -0.0355 | 2.7330 | 0.0560 |
| relevance_keyword_overlap | 1.2779 | 0.1977 | 2.3735 | 0.0138 | 2.2725 | 0.1461 |
| relevance_lexical_overlap (*) | 1.1371 | 0.2316 | 2.0148 | -0.0144 | 1.9182 | 0.1187 |
| relevance_rouge | 1.3079 | 0.2066 | 2.5603 | 0.0232 | 2.4838 | 0.1327 |
| relevance_semantic_sim_strong (*) | **0.8759** | **0.4092** | 1.1182 | 0.0395 | **0.9866** | 0.1601 |
| safety_toxicity (*) | 1.5396 | -0.0380 | **1.1105** | 0.0300 | 1.0139 | -0.0043 |
| structure_avg_paragraph_length_dist | 1.4560 | -0.1883 | 2.5562 | -0.0081 | 2.4637 | 0.1074 |
| structure_avg_sentence_length_dist | 1.5248 | -0.0140 | 2.4407 | -0.0287 | 2.4179 | 0.1612 |
| structure_cohesion_score | 1.4078 | 0.2070 | 2.7139 | 0.0345 | 2.6578 | 0.1567 |
| structure_emphasis_count | 1.2826 | 0.1988 | 2.6642 | 0.0482 | 2.5955 | 0.2060 |
| structure_headings | 1.4765 | 0.0423 | 2.6521 | -0.0340 | 2.5916 | 0.1049 |
| structure_lexical_diversity | 1.0672 | 0.1625 | 2.1864 | 0.0444 | 2.0981 | 0.1935 |
| structure_list_usage | 1.6284 | 0.0159 | 3.0208 | -0.0108 | 3.0132 | 0.0872 |
| structure_logical_transitions (*) | 1.2694 | 0.1743 | 2.2693 | 0.0520 | 2.4355 | 0.2263 |
| structure_max_sentence_length (*) | 1.3039 | 0.1272 | 2.7532 | 0.0104 | 2.7511 | 0.1377 |
| structure_min_sentence_length | 1.3568 | 0.1887 | 2.4872 | 0.0400 | 2.4322 | 0.2046 |
| structure_questions | 1.2443 | 0.2692 | 2.4910 | 0.0360 | 2.4064 | 0.2114 |
| structure_sentence_balance | 1.4423 | 0.1835 | 2.6757 | 0.0501 | 2.6444 | 0.2203 |
| structure_sentence_count (*) | 1.3099 | 0.1742 | 2.4408 | **0.0807** | 2.6570 | **0.2300** |

### E.5 Effects of Prompt-Based Intervention (Section 3.4)

We begin by analyzing how the intervention using a robust prompt affects the performance of individual LLM judges. Figures 5 (MAE) and 6 (Kendall's $\tau$) present the performance differences relative to the vanilla prompt. While the intervention aims to reduce confounding signals, its impact varies—some model–dataset combinations show improvement, while others show degradation.

We then assess how these shifts influence aggregate performance. Figures 7 and 8 show the corresponding changes in aggregation accuracy. Most baseline methods benefit from the intervention, whereas CARE shows a slight performance drop. A plausible explanation is that once confounding signals are mitigated, the additional latent variables in CARE may begin to model residual noise rather than meaningful structure, slightly reducing its performance. Nevertheless, as shown in Section 3.4, CARE without intervention still outperforms other baselines with the robustness prompt, highlighting its effectiveness even without manually crafted interventions for hidden confounders.

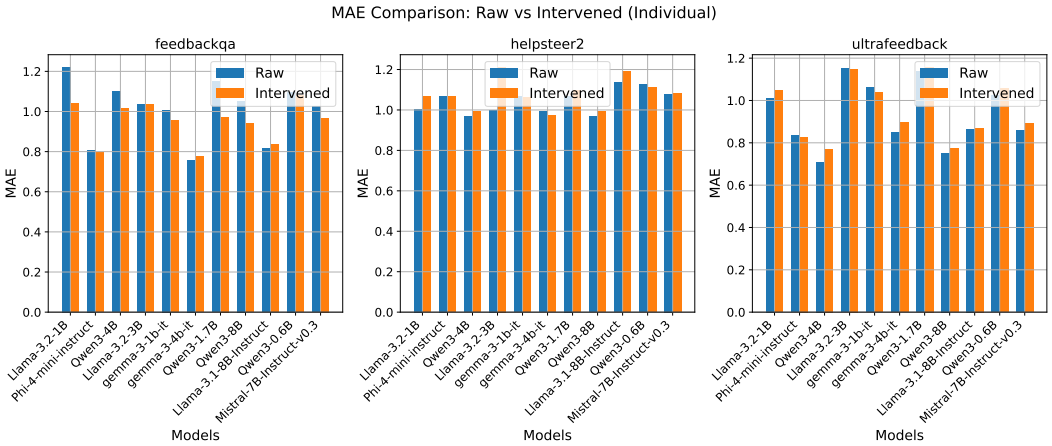



Figure 5: Change in MAE (↓) for individual LLM judges after applying the robustness prompt.

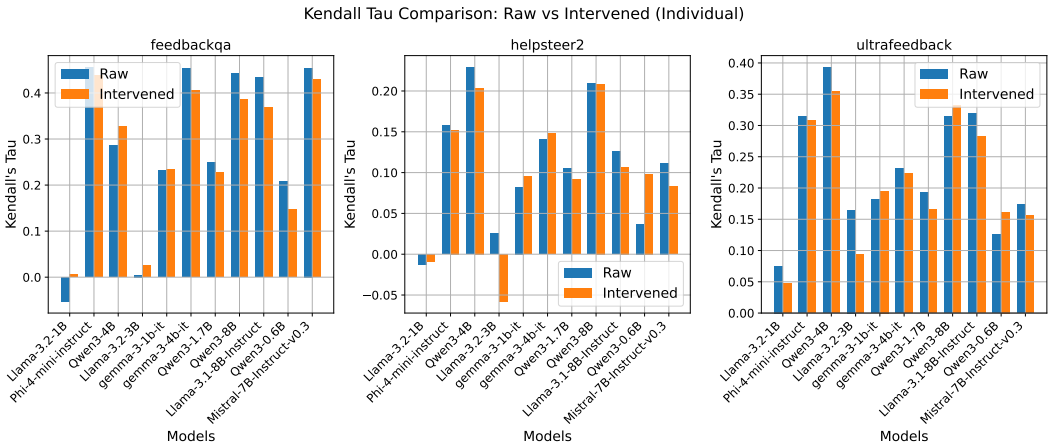



Figure 6: Change in Kendall's $\tau$ (↑) for individual LLM judges after the robustness prompt.

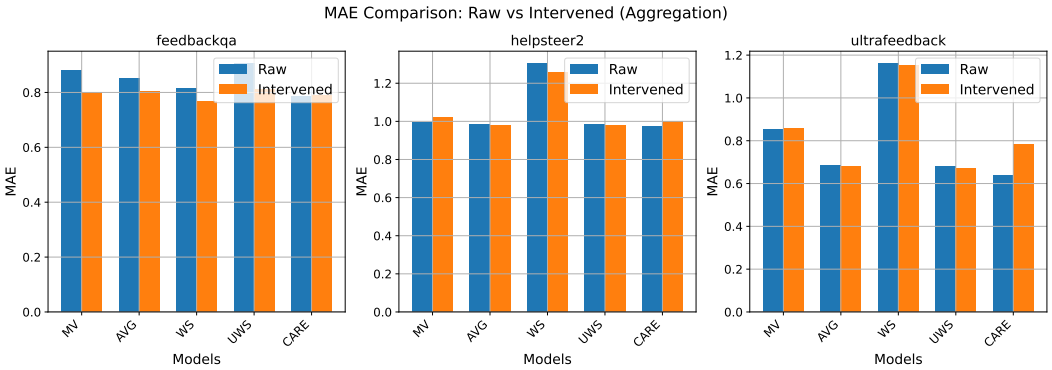



Figure 7: Change in aggregate MAE (↓) after propagating the robustness prompt through each aggregation method.

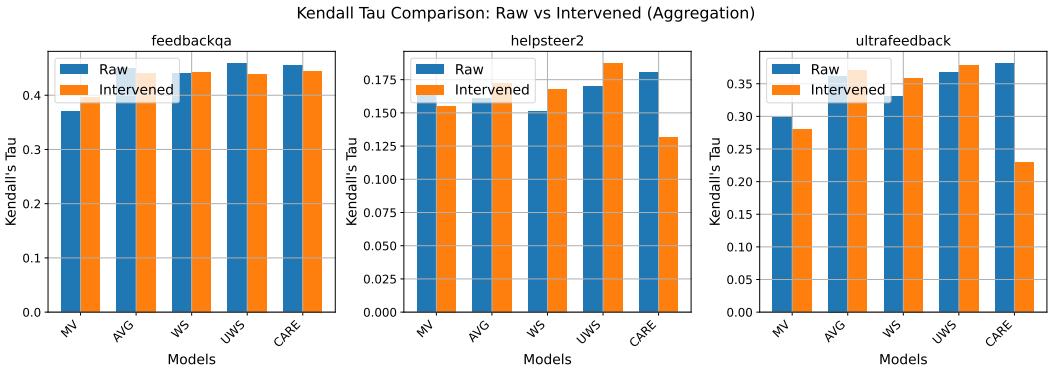



Figure 8: Change in aggregate Kendall's $\tau$ (↑) after the robustness prompt.

Table 7: Robustness to artificially injected bias. CARE is particularly effective against stylistic biases such as beauty (rich content) and authority, but less effective for gender and fallacy biases, which may impact the actual quality of system answers.

| | **Beauty Bias** | | **Fallacy Oversight Bias** | | **Gender Bias** | | **Authority Bias** | |
|---|---|---|---|---|---|---|---|---|
| | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) | MAE (↓) | $\tau$ (↑) |
| MV | 0.9190 | 0.3336 | 1.8971 | -0.0284 | 1.7428 | 0.1272 | 0.8239 | 0.2977 |
| AVG | 0.5063 | 0.3943 | **1.4007** | **0.1181** | **1.1355** | **0.2879** | 0.3250 | 0.4288 |
| WS | 1.9225 | 0.3792 | 2.5588 | 0.0680 | 2.0217 | 0.2474 | 0.9296 | 0.4886 |
| UWS | 0.5080 | 0.4383 | 1.3826 | 0.0491 | 1.1646 | 0.2576 | 0.2705 | 0.5799 |
| CARE | **0.3749** | **0.5334** | 1.8996 | 0.0116 | 1.5985 | 0.2311 | **0.2466** | **0.6327** |

### E.6 Robustness to Confounding Factors

**Setup.** We evaluate robustness using the dataset from [8], in which LLM responses are systematically altered to introduce specific biases via targeted GPT-4 prompts. The dataset includes four types of injected bias: *beauty*, *fallacy oversight*, *gender*, and *authority*. LLM judges are prompted to assign scores from 1 to 10 for each response. Robustness is assessed by comparing aggregated scores before and after bias injection, using mean absolute error (MAE) and Kendall's $\tau$. Lower MAE and higher Kendall's $\tau$ indicate better robustness under perturbation.

**Injected Confounders.** To clarify the setup, Table 8 summarizes the injected confounding factors with illustrative examples. These perturbations target different dimensions of bias, ranging from superficial stylistic changes to alterations that directly affect semantic correctness.

| **Bias** | **Perturbation Injected** | **Example Snippet (from Fig. 1)** |
|---|---|---|
| *Fallacy Oversight* | Insert a factual error | "The square root of 36 is **7**..." (correct value is 6) |
| *Authority* | Add a fake citation | "...(Weisstein, Eric W. 'Square Root.' MathWorld...)" |
| *Beauty* | Add emojis / formatting | "⑥ multiplied by ⑥ equals 36." |
| *Gender* | Add a gender-biased remark | "This might be a bit difficult for **women** to understand..." |

Table 8: Injected confounding factors and illustrative snippets.

**Results.** Table 7 reports the robustness of different aggregation methods under these injected biases. We find that CARE is highly stable against stylistic biases such as *beauty* and *authority*, preserving both rankings and score magnitudes. In contrast, robustness deteriorates when the bias directly undermines factual or semantic content—as in *fallacy oversight* and *gender* perturbations.

This distinction aligns with our hypothesis: *fallacy oversight* introduces factual inaccuracies that reduce answer quality, producing expected shifts in judge scores. Meanwhile, *gender* bias activates explicit safety mechanisms in alignment-tuned LLM judges, leading to consistent downscoring across models and correspondingly large shifts in aggregate outcomes.

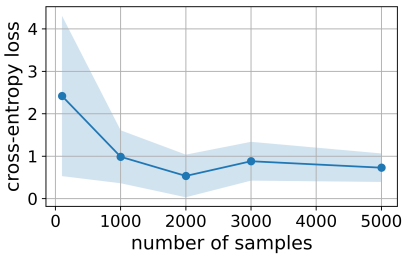



Figure 9: Averaged cross-entropy loss of our algorithm versus the number of samples. Markers denote average over three random seeds, and the shaded band denotes one standard deviation.

### E.7 Additional Controlled Experiment on Confounding Factors

Unlike the semi-synthetic perturbations in E.6, here we investigate whether CARE can separate the true quality latent factor from naturally arising confounders in a more controlled setting. Specifically, we introduce two dummy judges whose scores are directly correlated with response length or the presence of specific words. If CARE functions as intended, CARE should recover a factor structure in which high-quality judges align with the true quality factor $Q$, while the dummy judges align with a distinct confounder.

**Setup.** We ran CARE-SVD with 14 judges on the FEEDBACKQA dataset, combining 10 LLM judges, 2 programmatic "dummy" judges (sensitive to length or special keywords), and 2 human annotators. The factor loadings are presented in Table 9.

**Results.** The observed loadings align with our hypothesis:

- **Factor 1 (true quality $Q$).** This factor exhibits *broad, balanced loadings* across competent LLM judges and the two human judges, with much weaker loadings for the programmatic dummy judges. Within model families, larger models have higher loadings (e.g., Llama-3.1-8B $>$ Llama-3.2-3B $\approx$ Llama-3.2-1B), suggesting that $Q$ reflects underlying capability. Instruction-tuned models (Mistral-7B-Instruct, Phi-4-mini-instruct, Llama-3.1-8B-Instruct, Gemma-3-4B-it) also show above-median loadings, consistent with their alignment to human rubrics.
- **Factor 2 (length confounder).** This factor is dominated by a *high, concentrated loading* on the length-sensitive dummy_eval_1, with a secondary loading on gemma-3-1b-it (0.59). In contrast, nearly all other judges—including both humans and stronger instruction-tuned models—have near-zero loadings. Such a one-sided, few-judge pattern is characteristic of a confounder rather than true quality.

Table 9: Judge loadings on latent factors in CARE-SVD. Factor 1 corresponds to true quality $Q$; Factor 2 reflects a length confounder.

| **Judge** | $Q$ **(true quality)** | **Length confounder** |
|---|---|---|
| Qwen3-8B | 0.396 | -0.240 |
| Llama-3.1-8B-Instruct | 0.664 | -0.076 |
| gemma-3-4b-it | 0.706 | -0.152 |
| Llama-3.2-1B | -0.009 | -0.140 |
| Qwen3-4B | 0.180 | 0.008 |
| gemma-3-1b-it | 0.243 | 0.595 |
| Llama-3.2-3B | 0.033 | 0.057 |
| Phi-4-mini-instruct | 0.715 | -0.051 |
| Qwen3-1.7B | 0.199 | -0.012 |
| Mistral-7B-Instruct-v0.3 | 0.804 | 0.016 |
| dummy_eval_1 | 0.098 | 0.742 |
| dummy_eval_2 | 0.035 | 0.290 |
| human_eval_1 | 0.337 | 0.078 |
| human_eval_2 | 0.338 | 0.059 |

### E.8 Synthetic Experiments on Care-Tensor

We evaluate the performance of CARE-Tensor using simulated binary-Gaussian mixture data. Dataset details deferred to Appendix.

**Sample Complexity Result.** We investigate how the sample size $n$ influences estimation accuracy. We estimate conditional means $\hat{\mu}_{qc}$ and latent state proportions $\hat{\pi}_{qc}$ using Algorithm 3. Subsequently, we compute the posterior probabilities $P(Q = 1 \mid J)$ via the Bayesian formulation in Eq. 1. We measure the performance using cross-entropy loss. Lower entropy loss yields more accurate prediction. We observe a clear decreasing trend in cross-entropy loss as sample size increases.

**Tensor Decomposition vs SVD.** We illustrate the advantage of tensor decomposition over classical eigen-decomposition (SVD) in addressing rotation ambiguity with higher-order moments. We quantify performance using mean squared error (MSE) between true conditional means $\mu_{qc}$ and estimated means $\hat{\mu}_{qc}$. Detailed methodologies for SVD estimation are deferred to the appendix.

Evaluating across 10 random seeds, we find substantial performance differences: CARE-Tensor achieves significantly lower estimation errors with MSE ($\mathbf{0.51 \pm 0.41}$) compared to the eigen-decomposition baseline (SVD) with MSE ($\mathbf{1.18 \pm 0.74}$). This shows tensor decomposition accurately recovers conditional means without affected by rotation ambiguity.

### E.9 Synthetic Experiment on Graph-Aware Tensor Decomposition

When judges exhibit conditional dependencies, naively partitioning them into views violates the independence assumptions required by tensor decomposition. We hypothesize that partitioning judges via a graph-aware procedure that respects dependency structure yields substantially better estimation than random partitioning.

**Setup.** We simulated 10,000 items scored by $p = 12$ judges, partitioned into three views of four judges each. To induce conditional dependencies, we planted edges of strength $0.3$ within each true view at 40% density. We then compared two grouping strategies across ten random seeds:

- **Random**: assign judges to views uniformly at random;
- **Graph-Aware**: assign views to minimize cross-block edges in the empirical precision matrix.

Performance was measured by the $\ell_2$ error in recovering the latent component means, i.e. $||\mu_{qc} - \hat{\mu}_{qc}||_2$).

**Results.** As shown in Figure 10, graph-aware grouping dramatically reduces reconstruction error—by more than an order of magnitude—compared to random grouping. This confirms the importance of respecting dependency structure during view formation and underscores the advantage of CARE, which integrates graph structure directly into the tensor decomposition procedure.

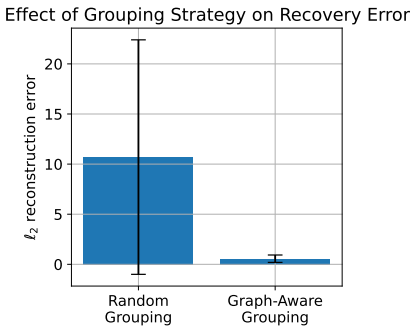
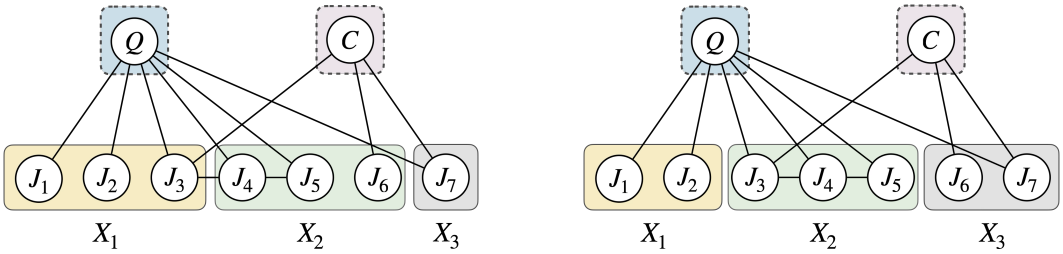



Figure 10: $\ell_2$ reconstruction error (mean $\pm$ SD) for random vs. graph-aware grouping.

### E.10 Additional Real-World Experiment on Gaussian Mixture

We consider a Gaussian mixture setting where the latent variable is binary, but the observables (judge outputs) are real-valued Gaussian scores. This experiment evaluates the effectiveness of Algorithm 3 on a real dataset.

**Setup.** We use a subset of the CivilComments dataset [52], randomly sample 5,000 examples. The ground-truth label is binary toxicity (0 or 1), while LLM judges provide real-valued toxicity scores ranging from 0 to 9. In addition to the original LLM judges, we include five LLMs:




(a) Random Partitioning (b) Graph-aware Partitioning

Figure 11: Random Partitioning vs. Graph Aware Partitioning. A random partitioning (a) leaves cross-view edges that violate the independence assumptions of tensor methods, whereas the graph-aware partitioning (b) considers cross-view edges and restores the required separation.

- `meta-llama/Meta-Llama-3-8B-Instruct`,
- `mistralai/Mistral-7B-Instruct-v0.2`,
- `Qwen/Qwen2.5-0.5B-Instruct`,
- `Qwen/Qwen2.5-1.5B-Instruct`,
- `Qwen/Qwen2.5-3B-Instruct`.

For the MV and WS baselines, we first discretize judge scores using a threshold of 4.5 before applying majority vote or weighted sum. For AVG and UWS, we aggregate scores first, then apply the threshold. CARE (Algorithm 3) directly infers the latent binary label from continuous scores. We evaluate all methods using classification accuracy.

Table 10: Aggregated accuracy (higher is better) in CivilComments dataset.

| **Method** | **Acc. (%)** |
|---|---|
| MV | 74.32% |
| AVG | 73.80% |
| WS | 74.95% |
| UWS | 74.95% |
| CARE | **75.27%** |

**Results.** Table 10 shows that CARE achieves the highest accuracy. This result highlights its ability to better handle confounding factors and perform effective latent inference, even when the observed data (continuous scores) differ from the latent variable type (binary labels).

## F Broader Impact Statement

This work presents a novel approach to aggregate scores from multiple LLMs serving as judges by identifying confounding variables and thus potentially reducing the bias in the overall judge scores. The potential broader impact includes a framework for improved LLM-as-a-judge scores which can be used at various applications. However, it is important to acknowledge that using LLMs as potential judges to automate labor-intense annotation tasks which is an active area of research carries some limitations and past research has discussed some unintended consequences, such as over-reliance on judge outputs, misuse and misinterpretation of results which might carry high real-world stakes. It is crucial to use automated LLM-as-a-judge tools responsibly and ethically, considering potential biases in data and models, and ensuring transparency and accountability in their application.