Automated Benchmarking of LLMs: Applying Regression to Estimate LLM Accuracy

Suryaansh Jain[a], **Umair Z. Ahmed**[b], **Shubham Sahai**[b], **Ben Leong**[b]

[a] *Indian Institute of Technology Hyderabad* cs21btech11057@iith.ac.in

[b] *National University of Singapore* umair@nus.edu.sg, shubham@nus.edu.sg, benleong@comp.nus.edu.sg

1. Introduction

The frequent release of new Large Language Models (LLMs) necessitates reliable and efficient benchmarking techniques to guide adoption decisions. Application developers have to constantly decide whether they should switch to a newer model or continue with an existing one. Different pricing structures and varying strengths across models further complicates this decision [1]. It is not enough to know whether one model is better over another [2]; developers need to know how much better it is to make the correct price-performance trade-off.

In this paper, we examine how to benchmark LLMs for a class of problem where many different answers could be correct. Human evaluation, while ideal, is too resource-intensive for it to be practical at scale. We need automated methods to reliably estimate correctness of LLM generated output.

Recent research has explored using LLMs themselves as evaluators for correctness. However, our experiments reveal a critical flaw with this approach: while LLMs excel at recognizing correct answers, they are surprisingly poor at identifying when other LLMs produce incorrect output, with their accuracy in identifying inaccurate output hovering around mere 25%. This finding has significant implications on using LLMs to benchmark LLMs.

To address this challenge, we introduce a novel regression-based approach that combines LLM evaluator predictions with a small set of human validated data to achieve more reliable benchmarking results at scale.

2. Related Work

Automatically evaluating LLMs on tasks with clear right or wrong answers, such as multiple-choice questions, is straightforward [3, 4]. The challenge arises when dealing with open-ended tasks where multiple different responses could be valid. Approximate matching techniques like BLEU [5] and ROUGE [6] attempt to address this by measuring similarity with reference answers. However, these approaches often fall short in aligning with human judgment, especially for tasks with multiple valid outputs that require deep semantic understanding [4].

The current state-of-the-art relies on using other LLMs as evaluators [7], typically through a pair-wise comparison of two different model outputs for the same task [8]. While studies show correlation between LLM-based and human evaluations [9], there are inherent reliability issues [10, 11]. Hence we need more accurate estimation methods for benchmarking that can be validated against human expert judgment.

Table 1: Predicted and actual precision of generators

	Validators		
Generators	**Mean**	**Ensemble**	**Human**
GPT 3.5T	73.0%	65.5%	69.6%
GPT-4	90.8%	91.8%	87.1%
GPT 4o-M	94.1%	95.2%	*
GPT 4o	90.9%	92.5%	93.5%
Opus 3	96.1%	97.0%	95.3%
Sonnet 3.5	96.6%	97.1%	*
G 1.5 flash	96.3%	97.5%	*
G 1.5 pro	96.3%	97.5%	92.9%
Qwen	95.3%	95.5%	92.8%
Deepseek	94.8%	95.4%	93.2%
Mean Error	2.6%	3.0%	–
Max Error	3.7%	4.7%	–

3. Baseline Approach

Recent works have explored the feasibility of using LLMs as validators [7, 8, 9], raising a fundamental question: *how accurate are LLMs at estimating the correctness of outputs produced by other models?*

Dataset. We used a dataset of 366 incorrect student programs from 69 diverse high school programming assignments [12]. The well-defined task being: determine whether the feedback output generated by LLM correctly addresses issues in the students' code (see Appendix A). We generated feedback on the incorrect student submissions using 10 LLMs and manually annotated the output of 7 of them (see Appendix B). Although manual annotation provides a reliable ground truth, it is unscalable due to being time-intensive task, which motivates the need for reliable automated evaluation methods.

LLM as Validators. Our findings reveal a surprising observation: LLM evaluators exhibit a strong positive bias. In other words, while LLMs are excellent at identifying valid outputs from other LLMs, achieving a True Positive Rate (v_j^+) of 93.5%, they perform poorly when it comes to recognizing invalid outputs, achieving a True Negative Rate (v_j^-) of only 25.2%. The results for the 10 LLM evaluators are provided in Appendix C.

This problem is masked in typical evaluations because there are far more correct outputs than incorrect ones, leading to an illusion of high overall accuracy. To the best of our knowledge, we are the first to highlight this critical limitation in LLM-based evalua-

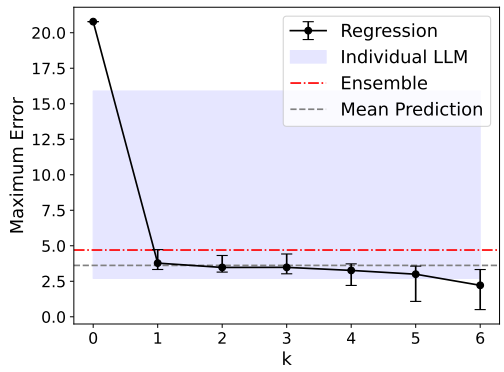

Fig. 1: Max error for estimating generator precision.

tion, raising concerns about the reliability of current automated benchmarking approaches.

Using an Ensemble of LLMs. A natural extension to address the limitations of using a single LLM for evaluation is to instead use an ensemble of LLMs [13, 2], where a majority vote determines whether an output is valid. However, given our observation that LLMs overestimate correctness and struggle to identify incorrect outputs, a naive majority-voting approach is unlikely to be effective.

To overcome this issue, we investigated a Simple Voting Invalid (SVI) approach, where an output is labeled invalid if at least n LLMs agree that it is incorrect. By allowing even a small minority of LLMs to classify an output as invalid, SVI compensates for the positive bias exhibited by LLMs and provides a more reliable way to detect invalid responses. Table 1 presents the predicted precision obtained using our ensemble-based approach. Notably, SVI with $n = 3$ (out of total 10 LLM validators) achieves a higher True Negative Rate of 30.4% when compared to individual LLMs. Which shows that while the "wisdom of the crowd" can help to improve performance, it is limited by the capability of the strongest LLMs in the pool. This raises the question: *Can we overcome this limitation and provide a more accurate estimate of the precision of a new model?*

4. Regression-based Approach

To addresses the fundamental limitation of LLMs being too agreeable, we propose a novel regression-based method. Instead of relying solely on LLMs to evaluate each other, our approach models the relationship between the true quality and LLM-perceived quality. Our solution is based on two key insights: (i) LLMs consistently overestimate the correctness of outputs, and (ii) we typically have access to at least some human-validated ground truth data for a few LLMs. By leveraging this limited human-validated data, we can calibrate LLM evaluations for improved accuracy estimates for new models.

Our approach uses a probabilistic model that accounts for both the generator's ability to produce valid outputs, and the validator's ability to identify valid and invalid outputs.

For a generator model, its precision is defined as the probability that generator produces a valid output, represented by g_i. For a validator model, we define True Positive Rate (v_j^+) and True Negative Rate (v_j^-) as the probability that the validator correctly identifies valid and invalid outputs, respectively. When a validator evaluates an output, the probability it will judge the output as valid is given by

$$g_i.v_j^+ + (1 - g_i).(1 - v_j^-)$$

In other words, either the output is valid and the validator identifies it correctly, or the output is invalid and the validator makes a mistake. By using prediction data from multiple validators, we use regression techniques to solve for the unknown probabilities that best explain the observed results. We refer the reader to Appendix D for more details.

Regularization. Our basic regression approach resulted in a systematic offset, with the estimated precision values tending to be higher than the true values. This is likely due to the bias that already exists within LLM validator predictions. In order to counter this bias, we incorporate a small amount of human-validated ground truth data as anchor points.

This is achieved by adding regularization terms to our regression loss function, where (a) one term minimizes the difference between estimated and known generator precision g_i, and (b) two terms are used to minimize the loss between estimated and actual validator TPR (v_j^+) and TNR (v_j^-). This loss function is detailed in Appendix D.

Results. In Figure 1, we compare the maximum cross-validation error in estimating generator precision between individual LLM evaluators (blue region), mean of all LLM evaluators (black dotted line), LLM evaluator ensemble (red dashed line), and regression using $k = 0, 1, 2, \ldots$ ground truth datasets (black solid line).

Our regression approach, with just one ground truth dataset provided ($k = 1$), reduces the maximum error from over 20% for basic regression ($k = 0$) to less than 5%, with the help of regularization. This result outperforms an ensemble of ten different state-of-the-art LLM evaluators. Furthermore, as we continue to increase the amount of ground truth data used for training, our testing error continues to decrease, outperforming the mean of all LLM evaluators starting from $k = 3$ and beyond. Our experiments reveal that with just a few sets of human-annotation for generator models (3 or more), we can dramatically improve the accuracy of our estimates for all the other unseen models.

5. Conclusion

We have uncovered a critical blind spot in how AI systems evaluate each other: LLMs struggle to detect incorrect outputs from other LLMs, which undermines current benchmarking approaches that rely on them. Our regression-based approach addresses this gap by combining the scalability of LLM evaluations with the accuracy of human judgment.

Acknowledgments

This research is supported by the National Research Foundation Singapore under the AI Singapore Programme (AISG Award No: AISG2-TC-2023-009-AICET).

References

[1] Huu Tan Mai, Chu Cuong Xuan, and Heiko Paulheim. Do LLMs really adapt to domains? An ontology learning perspective. *Lecture Notes in Computer Science*, 15231:126–143, 2025.

[2] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[3] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[6] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Meriem Boubdir, Edward Kim, Beyza Ermis, Sara Hooker, and Marzieh Fadaee. Elo Uncovered: Robustness and Best Practices in Language Model Evaluation. In *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 339–352, 2023.

[9] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

[10] Ruiyang Zhou, Lu Chen, and Kai Yu. Is LLM a Reliable Reviewer? A Comprehensive Evaluation of LLM on Automatic Paper Reviewing Tasks. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9340–9351, Torino, Italia, May 2024. ELRA and ICCL.

[11] Hui Wei, Shenghua He, Tian Xia, Andy Wong, Jingyang Lin, and Mei Han. Systematic evaluation of llm-as-a-judge in llm alignment tasks: Explainable metrics and diverse prompt templates. *arXiv preprint arXiv:2408.13006*, 2024.

[12] Shubham Sahai, Umair Z Ahmed, and Ben Leong. Improving the Coverage of GPT for Automated Feedback on High School Programming Assignments. In *NeurIPS'23 Workshop Generative AI for Education (GAIED). MIT Press, New Orleans, Louisiana, USA*, volume 46, 2023.

[13] Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. Merge, Ensemble, and Cooperate! A Survey on Collaborative Strategies in the Era of Large Language Models. *arXiv e-prints*, pages arXiv–2407, 2024.

[14] Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2000.

Appendix A. Problem Definition

We define task $t \in \mathcal{T}$ to be a task for which there exist well-defined solutions that can be deterministically classified as valid or invalid using an objective predefined criteria. The challenge is that for this set of tasks $\mathcal{T}$, there is effectively an unbounded set of solutions and there is no computationally efficient method to check that the solution is correct.

As a motivating example, consider the example of a typical CS1 (introductory programming) course, where given a problem description and test cases, and a buggy program written by students, we want to generate feedback to help students identify and fix their mistakes effectively. In this instance, our set of tasks, $\mathcal{T}$, comprises of the dataset of 366 incorrect Python program submissions spanning 69 different programming assignments [12]. The task t is defined as "given a student's incorrect code, description of the problem and a list of failing test cases, we want to generate *correct* feedback that will help a student fix the mistake(s) in the code."

Like Sahai et al. [12], we use a Large Language Model (LLM) $G_i \in \mathcal{G}$ (set of all available LLMs) to generate the solution for $t \in \mathcal{T}$. A human expert can manually classify the output o for task t as *valid* or *invalid*, represented by $H(t, o) = 1$ and $H(t, o) = 0$, respectively. If we annotate the output for all $t \in \mathcal{T}$,

we can determine the precision g_i of G_i for $\mathcal{T}$. However, human annotation is resource-intensive and not scalable. Our goal is to estimate g_x for a new generator $G_x \in \mathcal{G}$ *without requiring more human annotation.*

Appendix B. Manual Annotation

In their original data set, Sahai et al. [12] only annotated the outputs for LLMs GPT-3.5 Turbo (GPT-3.5T) and GPT-4. We have since used annotated the outputs for 5 more modern LLMs: (i) OpenAI GPT-4o, (ii) Anthropic Claude Opus 3; (iii) Google Gemini-1.5 pro; (iv) Qwen Coder Plus and (v) DeepSeek Chat. In each case, we provided the 366 tasks to each LLM, which generated appropriate feedback for the student. Given that we are dealing with novice programmers, the feedback provided is granular and tied to a particular line of code. Hence it is possible for an LLM to generate multiple lines of feedback for a given submission. Human experts then classified each line of feedback as valid or invalid to compute the precision g_i of each LLM as a generator G_i, which we present in Table A1.

Note that LLMs may sometimes fail to generate any feedback for some submissions, either due to complexity of the task or violation of expected output format. Except for GPT-3.5T, the number of failures is generally small and does not affect our results. We define the precision g_x to be the proportion of valid feedback within the set of feedback that were successfully generated by an LLM G_x. For now, we exclude the tasks for which G_x fails to generate valid output. The count of such failure cases are presented in Table A1 for reference.

Table A1 presents the manual annotation of our dataset. There are sub-categories among the valid or invalid feedback. In particular, valid feedback that either identify issues or suggest fixes are labeled as true positive (TP); feedback that propose improvements in code quality are labeled as true positive with extra suggestions (TP-E); and feedback that provide redundant information such as code explanation are labeled as true positive redundant (TP-R). Conversely, invalid feedback was categorized as either incorrect suggestions (FP-I) or hallucinated feedback (FP-H) that should not have been generated in the first place.

While human experts provide a reliable way to classify and evaluate feedback, manually analyzing them requires significant time, effort and domain expertise, making this approach inherently limited in scalability. It took us an estimated 200 man-hours to annotate the output for the 5 additional LLMs.

Appendix C. LLM as Validators

LLMs have been used by the community to validate the output of other LLMs [9]. So the first natural question is: *how good are LLMs in estimating the accuracy of other LLMs?*

Table A1: Feedback classification by human experts.

	Valid			Invalid		Precision
Generators	TP	TP-E	TP-R	FP-I	FP-H	g_i
GPT 3.5T*	263	61	5	66	78	69.6%
GPT-4*	633	123	8	75	38	87.1%
GPT 4o	734	188	24	54	12	93.5%
Opus 3	693	89	12	32	7	95.3%
G 1.5 pro	978	103	49	81	6	92.8%
Qwen	717	98	15	61	3	92.8%
Deepseek	793	160	20	49	22	93.2%

*original data set published at GAIED 2023 [12].

In addition to the 7 LLMs for which we have fully human-annotated outputs, we used 3 more LLMs (OpenAI GPT-4o-Mini, Anthropic Claude Sonnet 3.5 and Google Gemini-1.5 Pro) to generate feedback for our set of 366 buggy programs. Then we used all 10 LLMs to validate the output for all 10 LLMs. The results are shown in Table A2. In other words, Table A2 presents a view of how LLMs think of the outputs of other LLMs.

We make the following 2 observations from Table A2: first, the diagonal is not 100% as expected. In other words, sometimes an LLM surprisingly sometimes consider what it generated to be invalid. Second, the mean value across all the LLMs deviates quite substantially from the ground truth data for g_i. In particular, there seems to be a significant positive bias, i.e. LLMs generally over-estimate the correctness of other LLMs (except for GPT 4o).

Since we have the ground truth of the validity of the outputs for 7 LLMs, we investigated the accuracy of the various LLMs as validators (v_j). While the overall accuracy of the LLMs as validators was relatively high ($\approx$ 93%), we realized that the accuracy of the validation accuracy for valid and invalid outputs was substantially different. In particular, while the validation accuracy for valid outputs (v_j^+) was more than 90%, the validation accuracy for invalid outputs (v_j^-) was often below 20%!

To illustrate this issue, we have produced the corresponding True Positive Rate (v_j^+) and True Negative Rate (v_j^-) when the output for Opus 3 is validated by the various LLMs in Table A3. Due to space constraints, only results for Opus 3 output and the mean for all LLM outputs are shown.

The reason for the high overall accuracy in spite of the low v_j^- was because as we can see in Table A1, there is a much larger number of valid outputs (794) than invalid outputs (39), which skews the overall accuracy. Given this revelation, there is likely a need to revisit previous research using LLMs as validators to determine the impact of this phenomenon. In other words, an *LLM cannot easily tell when other LLMs make mistakes*. This observation seems hold for all 10 of the LLMs we studied.

Table A2 seems to suggest that Gemini-1.5 Pro produces the best estimate of the precision g_i. However, if we look at the accuracy of Gemini-1.5 Pro as a validator in Table A3, we see that the performance of

Table A2: Predicted precision of generators by validators $\mathrm{P'}_{ij}$, along with actual precision g_i assigned by humans

	Validators ($\mathcal{V}$)											
Generators	**GPT 3.5T**	**GPT 4T**	**GPT 4o-M**	**GPT 4o**	**Opus 3**	**Sonnet 3.5**	**G 1.5 flash**	**G 1.5 pro**	**Qwen**	**Deepseek**	**mean**	g_i
GPT 3.5T	79.8%	72.0%	75.2%	70.1%	70.3%	69.9%	80.5%	70.6%	66.9%	74.7%	73.0%	69.6%
GPT-4	87.7%	90.8%	90.2%	88.2%	90.6%	92.0%	95.3%	91.1%	87.7%	93.9%	90.8%	87.1%
GPT 4o-M	92.1%	94.4%	94.6%	94.1%	92.1%	94.6%	96.7%	94.0%	92.4%	96.2%	94.1%	
GPT 4o	92.1%	97.4%	83.6%	81.9%	77.6%	87.4%	97.9%	97.0%	96.1%	98.3%	90.9%	93.5%
Opus 3	93.2%	96.2%	95.3%	97.1%	97.8%	96.8%	97.8%	95.5%	94.2%	97.4%	96.1%	95.3%
Sonnet 3.5	93.6%	97.0%	96.1%	96.8%	95.8%	98.3%	98.0%	96.2%	96.0%	98.0%	96.6%	
G 1.5 flash	95.4%	94.5%	96.6%	95.8%	97.0%	95.6%	98.6%	96.7%	94.7%	97.8%	96.3%	
G 1.5 pro	94.3%	96.2%	95.4%	95.8%	96.8%	97.0%	97.6%	97.5%	95.3%	96.9%	96.3%	92.9%
Qwen	93.3%	96.1%	93.8%	95.6%	95.4%	96.1%	96.6%	93.7%	95.0%	97.7%	95.3%	92.8%
Deepseek	92.3%	95.1%	94.9%	95.6%	94.9%	93.7%	97.1%	94.7%	92.6%	96.8%	94.8%	93.2%
Min Error	0.5%	0.9%	0.0%	0.5%	0.7%	0.4%	2.5%	0.2%	0.6%	2.0%	0.8%	–
Mean Error	2.4%	2.8%	3.4%	3.3%	4.4%	3.0%	5.5%	2.2%	1.8%	4.5%	2.6%	–
Max Error	10.2%	3.9%	9.8%	11.6%	15.9%	6.1%	11.0%	4.6%	2.7%	6.7%	3.7%	–

Table A3: True Positive Rate (v_j^+) and True Negative Rate (v_j^-) of different LLM validators in predicting the *valid* and *invalid* label, respectively. We also report the overall (weighted) accuracy (v_j) for reference.

		Validators ($\mathcal{V}$)									
Generators		**GPT 3.5T**	**GPT 4T**	**GPT 4o-M**	**GPT 4o**	**Opus 3**	**Sonnet 3.5**	**G 1.5 flash**	**G 1.5 pro**	**Qwen**	**Deepseek**
Opus 3	v_j^+	93.4%	97.3%	96.6%	98.0%	98.5%	97.5%	98.3%	96.7%	95.5%	98.4%
	v_j^-	10.3%	27.8%	32.4%	21.6%	15.4%	17.9%	13.5%	28.2%	33.3%	23.1%
	v_j	89.4%	94.3%	93.7%	94.5%	94.6%	93.8%	94.5%	93.4%	92.6%	94.8%
mean	v_j^+	91.2%	94.8%	92.5%	92.1%	91.6%	93.9%	96.2%	93.9%	92.6%	95.9%
	v_j^-	14.8%	25.7%	31.6%	31.7%	29.1%	30.6%	14.1%	24.9%	29.1%	20.5%
	v_j	83.8%	88.7%	86.7%	86.8%	86.0%	88.5%	88.6%	87.8%	87.4%	89.0%

Gemini-1.5 Pro is only roughly 88% and comparable to the other LLMs. Delving into the data, what we found was that the apparent high accuracy for Gemini-1.5 Pro reflected in Table A2 could potentially be due to validation mistakes.

For instance, while validating the 833 lines of output generated by Claude Opus 3, we found that both GPT-4T and Gemini-1.5 Pro exhibited similar validation accuracy as shown in Table A3. However, the final precision estimates are different because of misclassification. In particular, GPT-4T correctly classifies 23 valid and 3 invalid inputs that Gemini-1.5 Pro misclassified; on the other hand, Gemini-1.5 Pro correctly identified 12 valid and 17 invalid outputs that GPT-4T misclassified. Some misclassifications will move reduce the estimation error, while other misclassifications might increase the error.

3.1 Using an Ensemble of LLMs.

Instead of using a single LLM to determine whether the output for an LLMs is correct, a natural approach used by many researchers is to use an ensemble of LLMs [13, 2] and do implement a voting mechanism, i.e. have an odd number of LLMs judge an outcome; if the majority agree that the answer is correct, then we conclude that the outcome is correct.

Given our observation that LLMs tend to have a very poor record at judging that other LLMs are wrong, a naive majority-vote-based scheme [2] will likely not to perform. Instead, we propose a modified scheme that consider the following 2 criteria:

1. at least m out of N LLMs agree that the output is valid (Simple Voting Valid); and
2. at least n out of N LLMs agree that the output is invalid (Simple Voting Invalid).

such that $1 \leq n < \frac{N}{2} \leq m \leq N$ and $n + m \leq N$. In some instances, both criteria would agree. Where the 2 criteria do not agree, an output is considered *invalid*. In other words, to mitigate the bias that LLMs have for agreeing with other LLMs, a small minority of the LLMs must be empowered to declare that an output is invalid.

Given that we have a set of human-annotated outputs (ground truth) and a set of LLMs $\mathcal{G}$, we can enumerate all possible values for n and m to determine the thresholds that yield the highest precision v_j for the available ground truth data set. We can also seek to minimize the harmonic mean of v_j^+ and v_j^- to address the skewness in the sample data.

In Table A4, we plot the predicted precision for different ensemble-based strategies. We also reproduce the ground truth precision g_i for the annotated outputs for reference. It is clear from these results that the pareto superior strategies are either $m = 7$ or $n = 4$ for naive voting strategies, or $m = 6, n = 4$ for the mixed strategy that we proposed. The difference between them will be a trade-off between the minimum, mean and maximum errors in the estimated precision values.

If we examine the validation accuracy of ensemble-based strategies in Table A5, we can clearly see that the True Negative Rates (v_j^-) for these 3 strategies are comparable and on average much higher than that for the individual LLMs. In fact, both $m = 7$ and $m =$

Table A4: Predicted precision of generators with ensemble-based strategy.

	Simple Voting Valid			Simple Voting Invalid			Both Criteria				
	$m=6$	$m=7$	$m=8$	$n=3$	$n=4$	$n=5$	$m=4$ $n=4$	$m=6$ $n=3$	$m=6$ $n=4$	$m=7$ $n=3$	g_i
GPT 3.5T	70.0%	63.8%	59.4%	61.7%	65.5%	73.2%	65.1%	61.3%	64.9%	61.1%	69.6%
GPT-4	93.2%	91.0%	87.1%	88.6%	91.8%	94.0%	91.8%	88.5%	91.6%	88.3%	87.1%
GPT 4o-M	95.0%	93.4%	91.9%	93.2%	95.2%	96.1%	94.3%	92.3%	94.3%	92.1%	
GPT 4o	97.0%	87.7%	79.2%	83.6%	92.5%	98.0%	92.5%	83.5%	92.0%	82.8%	93.5%
Opus 3	97.6%	96.5%	94.7%	96.0%	97.0%	97.8%	97.0%	96.0%	96.9%	95.9%	95.3%
Sonnet 3.5	95.9%	94.6%	93.5%	96.0%	97.1%	98.0%	95.2%	94.2%	95.2%	94.1%	
G 1.5 flash	95.4%	94.5%	92.0%	95.9%	97.5%	98.1%	95.2%	93.6%	95.2%	93.4%	
G 1.5 pro	97.9%	96.9%	94.2%	95.8%	97.5%	98.3%	97.5%	95.8%	97.5%	95.5%	92.9%
Qwen	96.1%	94.3%	91.9%	93.8%	95.5%	97.8%	95.3%	93.5%	95.2%	93.1%	92.8%
Deepseek	96.7%	93.4%	90.2%	93.6%	95.4%	97.2%	95.3%	93.5%	95.3%	92.0%	93.2%
Min Error	0.4%	0.2%	0.0%	0.4%	1.0%	2.5%	1.0%	0.3%	1.5%	0.2%	–
Mean Error	3.4%	3.2%	4.3%	3.5%	3.0%	4.5%	3.0%	3.5%	3.0%	3.6%	–
Max Error	6.0%	5.7%	14.2%	9.9%	4.7%	6.8%	4.7%	10.0%	4.7%	10.7%	–

Table A5: True Positive Rate (v_j^+) and True Negative Rate (v_j^-) of different LLM ensembles.

		Ensemble		
		$m=7$	$n=4$	$m=6 \oplus n=4$
All Outputs	v_j^+	92.6%	94.0%	92.6%
	v_j^-	34.7%	30.4%	34.7%
	v_j	87.9%	88.7%	87.9%

$6, n=4$ matches the best v_j^- for the individual LLMs (GPT-4o). This demonstrates that while the "wisdom of the crowd," can help to improve v_j^-, it is limited by the best among the available LLMs (31.7%). We need to do more to increase v_j^-.

Appendix D. Regression

To recap, given a new LLM G_x, we want to estimate the precision g_x without requiring additional human annotation. We had shown in §C.1 that an empirical ensemble-based approach is still not sufficiently accurate. We hence propose a novel regression-based approach that is based on 2 key insights: (i) existing LLMs tend to significantly over-estimate the correctness of invalid outputs; and (ii) we will generally have a set of annotated outputs for some subset of LLMs (i.e. we have the ground truth data for some outputs).

Validation Prediction Model. For generator $G_i \in \mathcal{G}$, we assume that it has a probability g_i of generating a valid output. For a validator $V_j \in \mathcal{V}$, we assume that it has a probability v_i^+ of validating a valid output correctly and a probability v_i^- of validating an invalid output correctly. Suppose V_j is used to validate the output for G_i, then the probability that V_j will declare that the output is correct will be $g_i \cdot v_j^+ + (1-g_i) \cdot (1-v_j^-)$. In other words, the generation was correct and the validation was correct, or the generation was wrong and the validation was also wrong.

Given this model and a set of probabilities g_i for all the generators $G_i \in \mathcal{G}$, and the sets of probabilities v_i^+ and v_i^- for the validators $V_j \in \mathcal{V}$, we can derive the expected prediction matrix $\widehat{\mathrm{P}}$ of dimension $|\mathcal{G}| \times |\mathcal{V}|$. Each cell P_{ij} represents the precision of generator G_i as measured by the validator V_j. We can also obtain the prediction matrix P directly by running experiments. In fact, the prediction matrix P for the 10 LLMs that we investigated is shown in Table A2.

Let $\widehat{g_i}$, $\widehat{v_j^+}$, and $\widehat{v_j^-}$ denote our estimates for the precision of generator i, True Positive Rate of validator j, and True Negative Rate of validator j, respectively. It remains for us to determine the set of probabilities $\widehat{g_i}$, $\widehat{v_j^+}$, and $\widehat{v_j^-}$ that will minimize the discrepancy between the observed values in P and the estimated values in $\widehat{\mathrm{P}}$. In other words, we have formulated our problem as a regression problem to estimate a total of $|\mathcal{G}| + 2|\mathcal{V}|$ variables using $|\mathcal{G}| \times |\mathcal{V}|$ values from P.

Loss Function. Given that we employ a form of gradient descent in our optimization and both P and $\widehat{\mathrm{P}}$ are class probabilities of the valid label, we employ the standard binary cross-entropy loss function $\mathcal{L}_{pred}$:

$$\mathcal{L}_{pred} = -\frac{1}{|\mathcal{G}||\mathcal{V}|} \sum_{i=1}^{|\mathcal{G}|} \sum_{j=1}^{|\mathcal{V}|} \mathrm{P}_{ij} \log \widehat{\mathrm{P}}_{ij} + (1-\mathrm{P}_{ij}) \log (1-\widehat{\mathrm{P}}_{ij}) \quad \text{(A1)}$$

Fixed Points. In practice, we observed that performing the minimization with $\mathcal{L}_{pred}$ alone still results in a systematic offset in precision estimates, i.e. the predicted precision values still tend to be higher than the true values. This is likely due to the inherent biases in empirical validator predictions. This is where we employ our second key insight: *we will generally already have available a set of ground truth data*. Instead of estimating the prediction matrix $\widehat{\mathrm{P}}$ without constraints, we can use known ground truth values in the loss minimization to reduce the effective number of free variables. In other words, instead of fitting $|\mathcal{G}| + 2|\mathcal{V}|$ variables using $|\mathcal{G}| \times |\mathcal{V}|$ values, we are fitting a reduced number of variables.

Hence, we modify the loss function by addition terms (similar to *regularization*) to minimize the dis-

Table A6: Mean error for estimating generator precision $\widehat{g}$.

k	0	1	2	3	4	5	6
Min	5.4%	1.8%	1.7%	1.4%	1.3%	0.6%	0.5%
Mean	5.4%	2.3%	2.3%	2.4%	2.3%	2.2%	2.2%
Max	5.4%	3.1%	3.0%	3.6%	3.1%	3.3%	3.3%

Table A7: Maximum error for estimating generator precision $\widehat{g}$.

k	0	1	2	3	4	5	6
Min	20.8%	3.3%	3.1%	3.0%	2.2%	1.1%	0.5%
Mean	20.8%	3.8%	3.5%	3.5%	3.3%	3.0%	2.2%
Max	20.8%	4.7%	4.3%	4.4%	3.7%	3.6%	3.3%

crepancies with known model parameters:

$$\begin{aligned}\mathcal{L}_{reg} = & \underbrace{\lambda_1\sqrt{\frac{1}{|\mathcal{H}|}\cdot\sum_{G_i\in\mathcal{H}}\left(g_i-\widehat{g}_i\right)^2}}_{\mathcal{L}_g} \\ & + \underbrace{\lambda_2\sqrt{\frac{1}{|\mathcal{V}|}\cdot\sum_{V_j\in\mathcal{V}}\left(v_j^+-\widehat{v}_j^+\right)^2}}_{\mathcal{L}_{v^+}} \\ & + \underbrace{\lambda_3\sqrt{\frac{1}{|\mathcal{V}|}\cdot\sum_{V_j\in\mathcal{V}}\left(v_j^--\widehat{v}_j^-\right)^2}}_{\mathcal{L}_{v^-}} \end{aligned} \tag{A2}$$

where $\mathcal{H}$ is a subset of LLMs for which we have access to the ground truth, such that $\mathcal{H}\subset\mathcal{G}$. We incorporate these corrections in our final loss function given by:

$$\mathcal{L} = \mathcal{L}_{pred} + \mathcal{L}_{reg}$$

where, $\mathcal{L}_{pred}$ and $\mathcal{L}_{reg}$ are formally defined in Equations (A1) and (A2) respectively. These additional terms in $\mathcal{L}_{reg}$ effectively penalize large deviations between the estimated probabilities and known ground truth probabilities.

Appendix E. Evaluation Results

In this section, we evaluate the accuracy of our regression-based approach for estimating generator precision g_i using our annotated data set for 7 LLMs. We compare its effectiveness against the performance of individual LLMs and best-case ensemble-based baselines.

5.1 Basic Regression

Our basic regression approach relies minimizing the prediction loss $\mathcal{L}_{pred}$ between the measured prediction matrix P and our model's estimated prediction matrix $\widehat{\mathrm{P}}$. In other words, we attempt to estimate $|\mathcal{G}|+2|\mathcal{V}|$ variables ($|\mathcal{G}|$ generator precision values $\widehat{g}_i$ and $2|\mathcal{V}|$ validator performance values $\widehat{v}_j^+$ and $\widehat{v}_j^-$) using $|\mathcal{G}|\times|\mathcal{V}|$ values from the measured prediction matrix P.

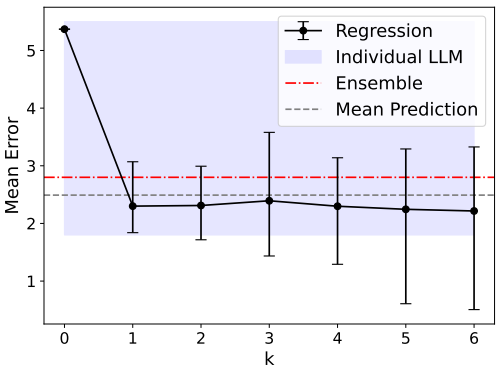

Fig. A1: Mean error for estimating generator precision $\widehat{g}$.

We use BFGS algorithm [14] with a default tolerance threshold of $\Delta\mathcal{L}_{pred} < 10^{-6}$. Generator precision values $\widehat{g}_i$ were initialized to the mean values of the predictions for the LLMs, while $\widehat{v}_j^+$ and $\widehat{v}_j^-$ were set to a neutral starting point of 0.5. The mean is obtained directly from the measured prediction matrix P. To ensure our results are robust against local minima, we conducted 10 experiment runs with random initialization offsets.

The performance of this naive formulation of the regression problem is very poor. As shown in Tables A6 and A7 (under $k=0$), the observed mean error was 5.4% and the maximum error was 20.8%. These results compare unfavorably to the use of individual LLMs (c.f. Table A2), or the use of ensembles.

5.2 Exploiting Ground Truth Information with $\mathcal{L}_{reg}$

If we have a set of the outputs for k LLMs, we can exploit this ground truth in an additional loss term $\mathcal{L}_{reg}$ (see Equation (A2)). In Tables A6 and A7, we plot the mean and maximum errors in the estimated precision values $\widehat{g}_i$, respectively. We have 7 annotated sets of LLM outputs. For $k=1$, we just use 1 set out of 7, but repeat the process for each set. Hence we have a range of possible values for each attempt; similar, for larger values of k, i.e. l, we repeat this process $\binom{k}{l}$ times to determine the range. We estimate the estimation error using the remaining annotated data sets, not used in the regression. This approach is not ideal, but annotated data is expensive and we only have limited sets of annotated data. The key insight in these experiments is that the use of just one data set (i.e. $k=1$) dramatically reduces both the mean and maximum error rates.

Tuning "Regularization" Constants (λ_1, λ_2, & λ_3). We use repeated fold cross-validation across $k\in[0,6]$, with different values of λ_i tuned using grid search. We found that higher weights for generator precision ($\lambda_1=10$) and True Negative Rate ($\lambda_3=10$) led to better estimates, compared to True Positive Rate weight ($\lambda_2=1$), which supports our key insight that accurately determining v_j^- values, will result in a better estimate of precision $\widehat{g}_i$.

5.3 Comparison with State of the Art

In Figures A1 and 1, we present the data in Tables A6 and A7 in a graphical format to allow for easy comparison with previous approaches. In particular, we consider the following three competing strategies:

1. use an individual LLM;
2. use the simple mean of all available (10) LLMs; and
3. use an ensemble-based strategy (§C.1).

If we pick an individual LLMs, both the mean and maximum errors can vary considerably as we can see in Table A2 and we are expected to operate in the blue region. Even if we pick what seems to be validator that achieves a low error rate, the good result could arise from random chance because v_i^- is relatively low (§3).

A simple mean of the predictions of all the LLMs (black dotted line) achieves surprisingly good performance. An ensemble-based strategy (§C.1) can improve the v_i^- and achieve comparable and consistent performance. The red dotted line in Figures A1 and 1 plot the best ensemble-based strategies for each case that we obtained by enumeration. Surprisingly, the error rates achieved with an ensemble is worse than that for the mean.

We see in Figures A1 and 1, that our regression-based technique can perform better than the state-of-the-art for $k \geq 1$. Performance is generally better if we use more ground truth data sets, i.e. if k is larger. In summary, our results demonstrate that even with a small set of ground truth annotations, we can effectively mitigate validator bias to significantly improve v_i^- estimates and thus the final result for our regression-based model. In particular, our results seem to suggest that we only need one or 2 sets of annotated LLM outputs, to achieve good results.

5.4 Discussion

Model Assumption. We have assumed in our model that each LLM validator has a fixed True Positive Rate (v_j^+) and True Negative Rate (v_j^-). When we investigate the v_j^+ and v_j^- for the various validators for different LLM output sets, we found that there are could be significant variations. However, if we did not make this simplifying assumption, then we would not be able to perform regression. Even if it were true that the there was true v_j^+ and v_j^-, the observed v_j^+ and v_j^- observed for different LLM output sets could also be different because the outputs could potentially be biased. The validity of the assumptions for our regression model requires further investigation. What is interesting is that our technique seems to work relatively well even though observed v_j^+ and v_j^- for different LLM output sets are different.

Impact of Prompt. We used the same prompts for both generation and validation for all LLMs. In particular, we have found that LLMs would occasionally not follow instructions. For example, they might not validate all the given lines, or worse, invent and suggest new lines. We have not had the opportunity to investigate the impact of the prompts used on both generation and validation accuracy. If a model seems to perform well or poorly, it could potentially be due to the prompt. Investigating the sensitivity of predictions and our regression technique to the prompts used and tuning prompts to improve validator output is left as future work.

Metric for Goodness. The current to estimate g_i by determining the validity of the LLM output naturally gives more weight to v_i^+. Instead of using g_i to determine the goodness of a new model, it might be helpful to consider the harmonic mean of v_i^+ and v_i^- instead. This would be similar to the F1 score.

Weighted Ensemble. Since some validators can be less reliable than others, a possible approach is to determine weights depended on the assessed reliability of each validator and using a weighted mean as the estimate. If we can weight LLMs with higher v_i^- more heavily, we could potentially improve overall v_i^-. This exploration is left for future work. We did not prioritize this approach because it does not address the low v_i^- problem directly and we did not feel such an approach could surpass the best performing LLM. Our current ensemble-based approach can already achieve on average the performance of the best performing LLM in the ensemble.

Correlation between Generation & Validation. We observed during our analysis that there was correlation between the validator's and generator's performance. In particular, when a generator struggled to produce an invalid output, and the validators also seem to have difficult validating the output correctly. For example, for the task of String checking, both LLM generators and validators struggled with escaping the string correctly. This suggests that difficulty of detecting invalid output is correlated to the difficulty of the task.