Uncertainty Quantification for Language Models: Standardizing and Evaluating Black-Box, White-Box, LLM Judge, and Ensemble Scorers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models often hallucinate, creating trust and safety concerns in high-stakes settings. We present a generation-time, zero-resource framework for hallucination detection that unifies black-box, white-box, and judge signals. Heterogeneous outputs are standardized to a shared 0 to 1 confidence scale for ranking and thresholding. To enhance flexibility, we introduce a simple, extensible ensemble with non-negative weights tuned on a graded set of LLM responses. Across six QA benchmarks and four generators, the ensemble outperforms the best individual scorer in 18 of 24 AUROC cases and 16 of 24 F1 cases. Among non-ensemble scorers, entailment-style black-box methods are strong baselines, although they incur higher generation costs and lack effectiveness when variation in sampled responses is low. The framework supports practical actions such as blocking low-confidence responses or routing to human review. We release an open-source Python library providing ready-to-use implementations of all methods.

1 Introduction

Large language models (LLMs) are increasingly deployed in production applications, including high-stakes domains such as healthcare and finance. Monitoring the accuracy and factuality of generated outputs is therefore essential. Hallucination, where outputs sound plausible yet contain incorrect content, remains a central reliability risk. Pre-deployment evaluations that compare generated text to ground truth are valuable but do not support real-time monitoring of deployed systems. In contrast, generation-time methods that quantify uncertainty can score each response at inference time.

Uncertainty quantification (UQ) spans black-box uncertainty from variation across sampled responses [Cole et al., 2023, Manakul et al., 2023, Lin et al., 2024, Kuhn et al., 2023, Kossen et al., 2024, Farquhar et al., 2024, Zhang et al., 2024, Qiu and Miikkulainen, 2024], white-box token-probability signals such as negative log probability, perplexity, entropy, and geometric means [Manakul et al., 2023, Fadeeva et al., 2024, Malinin and Gales, 2021], and LLM-as-a-judge scoring [Chen and Mueller, 2023, Kadavath et al., 2022, Xiong et al., 2024]. Surveys provide broader coverage [Huang et al., 2023, Tonmoy et al., 2024, Shorinwa et al., 2024, Huang et al., 2024]. Prior work often assesses a single family or considers heterogeneous signals without a shared output space.[1]

This paper presents a generation-time, zero-resource framework for hallucination detection that unifies diverse uncertainty signals. We adapt black-box methods that exploit response variation, white-box methods that use token probabilities, and LLM-as-a-judge methods. We then apply a standardization protocol that maps heterogeneous outputs to a common confidence scale in $[0, 1]$, where higher values

[1] We provide a detailed overview of related work in Appendix A.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

indicate higher confidence that a response is correct. The standardized scores enable ranking and thresholding for practical actions: block low-confidence responses, route uncertain cases to human review, or attach a low-confidence disclaimer. Building on these scores, we introduce a lightweight ensemble that combines multiple signals with non-negative weights learned on a graded set of LLM responses. The ensemble design is extensible, so new scorers can be added without redesign. Finally, our framework is complemented by `uqlm`, our open-source Python package that provides ready-to-use implementations of all uncertainty quantification methods presented and evaluated in this work.[2]

2 Hallucination Detection Methods

2.1 Problem Statement

We study binary detection of hallucinations in LLM outputs. For a prompt x_i and its original response $y_i \in \mathcal{Y}$, let a *confidence scorer* be a function $\hat{s} : \mathcal{Y} \to [0, 1]$ that maps y_i to a confidence score, where larger values indicate higher confidence that y_i is correct. Given a threshold $\tau \in [0, 1]$, define the binary predictor $\hat{h} : \mathcal{Y} \to \{0, 1\}$ by $\hat{h}(y_i; \theta, \tau) = \mathbb{I}(\hat{s}(y_i; \theta) < \tau)$, where θ denotes scorer-specific inputs (e.g., multiple responses generated from x_i).

2.2 UQ-Based Confidence Scorers

We group scorers into three families. Black-box scorers exploit variation across multiple responses to the same prompt. White-box scorers use token probabilities associated with the generated response. Judge-based scorers prompt an LLM to rate the correctness of a question-response concatenation. All raw outputs are standardized to $[0, 1]$ so that larger values indicate higher confidence. Standardization applies direction alignment and a bounded monotone transform when needed. Concise scorer definitions are presented in Table 1 and detailed definitions are contained in Appendix B. A detailed description of the `uqlm` software is contained in Appendix C.

Scorer	Family	Definition (sketch)	Extra inputs
EMR	Black-box	$\frac{1}{m}\sum_{j=1}^{m} \mathbb{I}(y_i = \tilde{y}_{ij})$	m candidates
BSC	Black-box	$\frac{1}{m}\sum_{j=1}^{m} \mathrm{BERTF1}(y_i, \tilde{y}_{ij})$	m candidates, token-level embeddings
NCS	Black-box	$\frac{1}{m}\sum_{j=1}^{m} \frac{1+\cos(V(y_i), V(\tilde{y}_{ij}))}{2}$	m candidates, sentence-level embeddings
NCP	Black-box	$1 - \frac{1}{m}\sum_{j=1}^{m} \frac{\eta(y_i, \tilde{y}_{ij}) + \eta(\tilde{y}_{ij}, y_i)}{2}$	m candidates, NLI model
NSN	Black-box	$1 - \mathrm{SE}(y_i; \tilde{\mathbf{y}}_i)/\log(m{+}1)$	m candidates, NLI model
LNTP	White-box	$\prod_{t \in y_i} p_t^{1/L_i}$	token probabilities
MTP	White-box	$\min_{t \in y_i} p_t$	token probabilities
Judge	LLM-as-judge	$\mathrm{score}_{\text{judge}}(x_i, y_i)/100$	judge model
Ensemble	Meta	Optimized combination of scorers	tuning data, component inputs

Table 1: Scorers at a glance: standardized $[0, 1]$ confidence; higher implies lower hallucination risk.

Black-Box UQ. For a prompt x_i, black-box UQ methods generate m candidates $\tilde{\mathbf{y}}_i = \{\tilde{y}_{i1}, \ldots, \tilde{y}_{im}\}$ at nonzero temperature and compare to the original response y_i. Methods include exact-match rate (EMR) [Cole et al., 2023, Chen and Mueller, 2023], semantic similarity via BERTScore confidence (BSC) [Manakul et al., 2023] or normalized cosine similarity (NCS) [Shorinwa et al., 2024], and entailment-based scorers including non-contradiction probability (NCP) [Chen and Mueller, 2023, Lin et al., 2024, Manakul et al., 2023] and semantic entropy [Farquhar et al., 2024, Kuhn et al., 2023] normalized to confidence, i.e., normalized semantic negentropy (NSN). These require additional generations and, for some variants, an encoder or a Natural Language Inference (NLI) model.

White-Box UQ. White-box UQ methods use token probabilities of the generated response to measure uncertainty. These approaches have the advantage of adding no additional latency or

[2] An anonymized GitHub repository with code and instructions is available at `https://anonymous.4open.science/r/uqlm-FBF3`. A public link will be provided upon acceptance.

generation costs, but are not compatible with all LLM APIs. We consider two white-box UQ scorers: length-normalized token probability (LNTP), computed as the geometric mean of token probabilities for a generated response, and minimum token probability (MTP). By construction, these two scorers are bounded with $[0, 1]$ support and hence do not require normalization.

LLM-as-a-judge. In this approach, we concatenate a question-response pair and pass it to an LLM with a carefully constructed instruction prompt that directs the model to evaluate the correctness of the response. We adapt our instruction prompt from Xiong et al. [2024], instructing the LLM to score responses on a 0-100 scale, where a higher score indicates a greater certainty that the provided response is correct. These scores are then normalized to a 0-to-1 scale to maintain consistency with our other confidence scoring methods. The complete prompt template is provided in Appendix B.4.

Ensemble (meta-scorer). Lastly, we combine standardized scorers using non-negative weights learned on a graded set of LLM responses to optimize a chosen objective (e.g., AUROC). The ensemble outputs a single $[0, 1]$ confidence score and remains interpretable; non-negative weights preserve monotonicity with respect to each component. Tuning details appear in Appendix B.5.

3 Experiments

Experimental Setup We evaluate hallucination detection across six QA benchmarks with diverse answer formats: numerical (*GSM8K* [Cobbe et al., 2021], *SVAMP* [Patel et al., 2021]), multiple-choice (*CSQA* [Talmor et al., 2022], *AI2-ARC* [Clark et al., 2018]), and open-text (*PopQA* [Mallen et al., 2023], *NQ-Open* [Lee et al., 2019]). We sample 1,000 questions per dataset and, for each prompt, generate one original response and $m{=}15$ candidates at temperature 1.0 using four LLMs: GPT-3.5 [OpenAI], GPT-4o [OpenAI], Gemini-1.0-Pro [Google], and Gemini-1.5-Flash [Google]. Black-box scores for a response use the candidates produced by the same LLM. Judge scores are obtained from three judges (GPT-3.5, GPT-4o, Gemini-1.5-Flash); white-box scores are computed where token probabilities are available (GPT-4o, Gemini-1.0-Pro, Gemini-1.5-Flash).[3] All scores are produced with our companion toolkit, `uqlm`.[4] Supplemental figures displaying detailed experiment results are contained in Appendix D.

Threshold-agnostic (AUROC). We evaluate each scorer as a ranker of incorrect responses using AUROC. The ensemble uses AUROC-optimized weights learned on the training fold with 5-fold cross-validation. Appendix Fig. 5 shows per-scorer AUROC across the 24 LLM–dataset scenarios, while Table 2 highlights the best scorer per scenario. Best-scorer AUROC ranges from 0.72 (GPT-3.5 on PopQA, NCS) to 0.93 (Gemini-1.5-Flash on AI2-ARC, LNTP), with 19 of 24 scenarios above 0.8, indicating strong overall hallucination detection performance. The ensemble outperforms individual components in 18 of 24 scenarios. NLI-based scorers (NSN, NCP) typically lead among black-box scorers (18 of 24 scenarios), GPT-4o is the strongest judge in 19 of 24 scenarios, and the two white-box scorers (LNTP and MTP) perform similarly.

Table 2: Hallucination Detection AUROC (Higher is Better): Top Scorer by *Dataset* and *Model*

Dataset	**Gem.-1.0-Pro**		**Gem.-1.5-Flash**		**GPT-3.5**		**GPT-4o**	
	AUROC	Scorer	AUROC	Scorer	AUROC	Scorer	AUROC	Scorer
NQ-Open	0.84	Ensemble	0.79	Ensemble	0.76	Ensemble	0.73	Ensemble
PopQA	0.86	Ensemble	0.87	Ensemble	0.72	NCS	0.91	Ensemble
GSM8K	0.84	Ensemble	0.82	Ensemble	0.84	Ensemble	0.90	Ensemble
SVAMP	0.88	NSN	0.89	MTP	0.88	Ensemble	0.89	Ensemble
CSQA	0.87	GPT-4o	0.79	Ensemble	0.84	Ensemble	0.84	Ensemble
AI2-ARC	0.90	Ensemble	0.93	LNTP	0.91	Ensemble	0.86	LNTP

[3] Our GPT-3.5 instance did not expose token probabilities. Gemini-1.0-Pro was retired during the study and is unavailable as a judge.

[4] Using an `n1-standard-16` machine (16 vCPU, 8 core, 60 GB memory) with a single NVIDIA T4 GPU attached, our experiments took approximately 0.5-3 hours per LLM-dataset combination to complete.

Threshold-optimized (F1). We jointly tune ensemble weights and the decision threshold with F1 as the objective (per-scorer thresholds via grid search), reporting 5-fold cross-validated test F1. Appendix Fig. 6 shows per-scorer F1, while Table 3 summarizes the top scorer per scenario. Results mirror AUROC: the ensemble is best in 16 of 24 scenarios, NSN/NCP dominate among black-box scorers (19 of 24), and GPT-4o is the strongest judge (22 of 24). GPT-4o's judging strength aligns with its higher baseline answer accuracy relative to GPT-3.5 and Gemini models (Appendix Fig. 7).

Table 3: Hallucination Detection F1 (Higher is Better): Top Scorer by *Dataset* and *Model*

Dataset	**Gem.-1.0-Pro**		**Gem.-1.5-Fl.**		**GPT-3.5**		**GPT-4o**	
	F1	Scorer	F1	Scorer	F1	Scorer	F1	Scorer
NQ-Open	0.62	EMR	0.67	Ensemble	0.68	Ensemble	0.75	Ensemble
PopQA	0.54	NSN	0.66	GPT-4o	0.42	Ensemble	0.75	Ensemble
GSM8K	0.63	Ensemble	0.70	NCP	0.60	Ensemble	0.85	Ensemble
SVAMP	0.89	NSN	0.93	NCP	0.89	Ensemble	0.97	Ensemble
CSQA	0.89	GPT-4o	0.90	Ensemble	0.90	Ensemble	0.91	NCP
AI2-ARC	0.97	Ensemble	0.97	Ensemble	0.97	Ensemble	0.99	Ensemble

Filtered Accuracy@τ. We report accuracy on the subset of responses with confidence $\geq \tau$ for $\tau \in \{0, 0.1, \ldots, 0.9\}$. Appendix Fig. 7 plots the best white-box, black-box, judge, and ensemble per scenario. Accuracy generally increases with τ. For example, filtering Gemini-1.0-Pro on PopQA with the top black-box scorer raises accuracy from 0.15 to 0.69 at τ=0.6; filtering GPT-4o on GSM8K with a white-box scorer yields 0.81 at τ=0.6 vs 0.55 baseline. A notable exception is black-box scoring on Gemini-1.5-Flash for CSQA and AI2-ARC, where filtering does not improve accuracy.

Effect of Candidate Count (m) on Black-Box Scorers We recompute black-box scores for $m \in \{1, 3, 5, 10, 15\}$ across 24 LLM–dataset scenarios (holding the original response fixed) and report AUROC (Fig. 8). Performance generally rises with m with diminishing returns [Kuhn et al., 2023, Manakul et al., 2023, Lin et al., 2024, Farquhar et al., 2024]. We find two exceptions: (i) BSC shows weaker gains with larger m, as also observed by Manakul et al. [2023]; (ii) for Gemini-1.5-Flash on AI2-ARC and CSQA, black-box AUROC is low (0.46–0.56) and flat because candidates are near-duplicates (EMR=1.00, 0.99). More broadly, Gemini-1.5-Flash exhibits higher EMR than other LLMs, underscoring that black-box effectiveness is limited by candidate diversity [Kuhn et al., 2023].

4 Conclusion

Discussion and Conclusion. Selecting a confidence scorer depends on API access, latency, model behavior, and the availability of a small graded set for tuning. If token probabilities are available, white-box scorers add no latency or generation costs and are competitive. Without token probabilities, black-box and judge-based scorers are the practical options. For low-latency applications, prefer faster black-box methods or a judge; when latency is less constrained, NLI-based black-box scorers (NSN, NCP) typically perform best. Black-box methods can struggle when sampled responses lack diversity. In such cases, white-box or judge-based signals are preferable (Fig. 8). Increasing the number of candidates m offers limited gains once diversity saturates, which provides a budget-aware guideline for black-box settings. For judge selection, a simple heuristic works well in our experiments: models with higher task accuracy tend to be better judges of answers on that task. When a graded set of LLM responses is available, a tuned ensemble over standardized scorers improves robustness over individual methods and remains extensible as new scorers appear. Practically, standardized scores enable thresholded actions: block low-confidence responses, route uncertain cases to human review, or attach a low-confidence disclaimer, with efficacy demonstrated in the filtered accuracy experiments (Fig. 7).

Limitations and Future Work. While our evaluation covers six QA datasets and four LLMs, it is important to note that results may differ for different question types (e.g. long-form tasks), newer models, or alternative judge prompts. White-box behavior depends on token-probability availability, and black-box effectiveness depends on response diversity. Our ensemble experiments are limited to linear ensembles. For future work, we suggest exploring richer ensembling strategies, broader task families, and more recently released LLMs in experimental evaluations.

References

A. Agrawal, M. Suzgun, L. Mackey, and A. T. Kalai. Do language models know when they're hallucinating references?, 2024. URL https://arxiv.org/abs/2305.18248.

Arize AI. Phoenix, 2025. URL https://github.com/Arize-ai/phoenix.

A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Y. F. Bakman, D. N. Yaldiz, B. Buyukates, C. Tao, D. Dimitriadis, and S. Avestimehr. Mars: Meaning-aware response scoring for uncertainty estimation in generative llms, 2024. URL https://arxiv.org/abs/2402.11756.

S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909/.

J. Chen and J. Mueller. Quantifying uncertainty in answers from any language model and enhancing their trustworthiness, 2023. URL https://arxiv.org/abs/2308.16175.

I.-C. Chern, S. Chern, S. Chen, W. Yuan, K. Feng, C. Zhou, J. He, G. Neubig, P. Liu, et al. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*, 2023. doi: 10.48550/arXiv.2307.13528.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

R. Cohen, M. Hamri, M. Geva, and A. Globerson. Lm vs lm: Detecting factual errors via cross examination, 2023. URL https://arxiv.org/abs/2305.13281.

J. R. Cole, M. J. Q. Zhang, D. Gillick, J. M. Eisenschlos, B. Dhingra, and J. Eisenstein. Selectively answering ambiguous questions, 2023. URL https://arxiv.org/abs/2305.14613.

S. Es, J. James, L. Espinosa-Anke, and S. Schockaert. Ragas: Automated evaluation of retrieval augmented generation, 2023. URL https://arxiv.org/abs/2309.15217.

E. Fadeeva, R. Vashurin, A. Tsvigun, A. Vazhentsev, S. Petrakov, K. Fedyanin, D. Vasilev, E. Goncharova, A. Panchenko, M. Panov, T. Baldwin, and A. Shelmanov. LM-polygraph: Uncertainty estimation for language models. In Y. Feng and E. Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 446–461, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.41. URL https://aclanthology.org/2023.emnlp-demo.41.

E. Fadeeva, A. Rubashevskii, A. Shelmanov, S. Petrakov, H. Li, H. Mubarak, E. Tsymbalov, G. Kuzmin, A. Panchenko, T. Baldwin, P. Nakov, and M. Panov. Fact-checking the output of large language models via token-level uncertainty quantification, 2024. URL https://arxiv.org/abs/2403.04696.

S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, Jun 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07421-0. URL https://doi.org/10.1038/s41586-024-07421-0.

Google. URL https://cloud.google.com/vertex-ai/generative-ai/docs/models.

N. M. Guerreiro, E. Voita, and A. F. T. Martins. Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation, 2023. URL https://arxiv.org/abs/2208.05309.

X. Hu, D. Ru, L. Qiu, Q. Guo, T. Zhang, Y. Xu, Y. Luo, P. Liu, Y. Zhang, and Z. Zhang. Refchecker: Reference-based fine-grained hallucination checker and benchmark for large language models, 2024. URL https://arxiv.org/abs/2405.14486.

H.-Y. Huang, Y. Yang, Z. Zhang, S. Lee, and Y. Wu. A survey of uncertainty estimation in llms: Theory meets practice, 2024. URL https://arxiv.org/abs/2410.15326.

L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023. URL https://arxiv.org/abs/2311.05232.

J. Ip and K. Vongthongsri. deepeval, Mar. 2025. URL https://github.com/confident-ai/deepeval.

M. Jiang, Y. Ruan, P. Sattigeri, S. Roukos, and T. Hashimoto. Graph-based uncertainty metrics for long-form language model outputs, 2024. URL https://arxiv.org/abs/2410.20783.

S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

J. Kossen, J. Han, M. Razzak, L. Schut, S. Malik, and Y. Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms, 2024. URL https://arxiv.org/abs/2406.15927.

L. Kuhn, Y. Gal, and S. Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. URL https://arxiv.org/abs/2302.09664.

K. Lee, M.-W. Chang, and K. Toutanova. Latent retrieval for weakly supervised open domain question answering, 2019. URL https://arxiv.org/abs/1906.00300.

C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Z. Lin, S. Trivedi, and J. Sun. Generating with confidence: Uncertainty quantification for black-box large language models, 2024. URL https://arxiv.org/abs/2305.19187.

C. Ling, X. Zhao, X. Zhang, W. Cheng, Y. Liu, Y. Sun, M. Oishi, T. Osaki, K. Matsuda, J. Ji, G. Bai, L. Zhao, and H. Chen. Uncertainty quantification for in-context learning of large language models, 2024. URL https://arxiv.org/abs/2402.10189.

Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

J. Luo, C. Xiao, and F. Ma. Zero-resource hallucination prevention for large language models, 2023. URL https://arxiv.org/abs/2309.02654.

A. Malinin and M. Gales. Uncertainty estimation in autoregressive structured prediction, 2021. URL https://arxiv.org/abs/2002.07650.

A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

P. Manakul, A. Liusie, and M. J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models, 2023. URL https://arxiv.org/abs/2303.08896.

OpenAI. URL https://platform.openai.com/docs/models.

OpenAI. Evals, 2024. URL https://github.com/openai/evals.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

A. Patel, S. Bhattamishra, and N. Goyal. Are nlp models really able to solve simple math word problems?, 2021. URL https://arxiv.org/abs/2103.07191.

X. Qiu and R. Miikkulainen. Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space, 2024. URL https://arxiv.org/abs/2405.13845.

A. W. Qurashi, V. Holmes, and A. P. Johnson. Document processing: Methods for semantic text similarity analysis. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6, 2020. doi: 10.1109/INISTA49547.2020.9194665.

T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In Y. Feng and E. Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.40. URL https://aclanthology.org/2023.emnlp-demo.40.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. URL https://arxiv.org/abs/1908.10084.

J. Ren, Y. Zhao, T. Vu, P. J. Liu, and B. Lakshminarayanan. Self-evaluation improves selective generation in large language models, 2023. URL https://arxiv.org/abs/2312.09300.

O. Shorinwa, Z. Mei, J. Lidard, A. Z. Ren, and A. Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions, 2024. URL https://arxiv.org/abs/2412.05563.

A. Talmor, O. Yoran, R. L. Bras, C. Bhagavatula, Y. Goldberg, Y. Choi, and J. Berant. Commonsenseqa 2.0: Exposing the limits of ai through gamification, 2022. URL https://arxiv.org/abs/2201.05320.

S. M. T. I. Tonmoy, S. M. M. Zaman, V. Jain, A. Rani, V. Rawte, A. Chadha, and A. Das. A comprehensive survey of hallucination mitigation techniques in large language models, 2024. URL https://arxiv.org/abs/2401.01313.

UpTrain AI Team. UpTrain, 2024. URL https://github.com/uptrain-ai/uptrain.

L. van der Poel, R. Cotterell, and C. Meister. Mutual information alleviates hallucinations in abstractive summarization, 2022. URL https://arxiv.org/abs/2210.13210.

N. Varshney, W. Yao, H. Zhang, J. Chen, and D. Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation, 2023. URL https://arxiv.org/abs/2307.03987.

P. Verga, S. Hofstatter, S. Althammer, Y. Su, A. Piktus, A. Arkhangorodsky, M. Xu, N. White, and P. Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models, 2024. URL https://arxiv.org/abs/2404.18796.

B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, C. Zhang, C. Xu, Z. Xiong, R. Dutta, R. Schaeffer, S. T. Truong, S. Arora, M. Mazeika, D. Hendrycks, Z. Lin, Y. Cheng, S. Koyejo, D. Song, and B. Li. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models, 2024. URL https://arxiv.org/abs/2306.11698.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

WhyLabs. langkit, 2025. URL https://github.com/whylabs/langkit.

M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms, 2024. URL https://arxiv.org/abs/2306.13063.

Y. Zha, Y. Yang, R. Li, and Z. Hu. AlignScore: Evaluating factual consistency with a unified alignment function. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.634. URL https://aclanthology.org/2023.acl-long.634/.

C. Zhang, F. Liu, M. Basaldella, and N. Collier. Luq: Long-text uncertainty quantification for llms, 2024. URL https://arxiv.org/abs/2403.20279.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

T. Zhang, L. Qiu, Q. Guo, C. Deng, Y. Zhang, Z. Zhang, C. Zhou, X. Wang, and L. Fu. Enhancing uncertainty-based hallucination detection with stronger focus, 2023. URL https://arxiv.org/abs/2311.13230.

A Related Work

Black-Box UQ Cole et al. [2023] propose evaluating similarity between an original response and candidate responses using exact match-based metrics. In particular, they propose two metrics: repetition, which measures the proportion of candidate responses that match the original response, and diversity, which penalizes a higher proportion of unique responses in the set of candidates. These metrics have the disadvantage of penalizing minor phrasing differences even if two responses have the same meaning. Text similarity metrics assess response consistency in a less stringent manner. Manakul et al. [2023] propose using n-gram-based evaluation to evaluate text similarity. Similar metrics such as ROUGE [Lin, 2004], BLEU [Papineni et al., 2002], and METEOR [Banerjee and Lavie, 2005] have also been proposed [Shorinwa et al., 2024]. These metrics, while widely adopted, have the disadvantage of being highly sensitive to token sequence orderings and often fail to detect semantic equivalence when two texts have different phrasing. Sentence embedding-based metrics such as cosine similarity [Qurashi et al., 2020], computed using a sentence transformer such as Sentence-Bert [Reimers and Gurevych, 2019], have also been proposed [Shorinwa et al., 2024]. These metrics have the advantage of being able to detect semantic similarity in a pair of texts that are phrased differently. In a similar vein, Manakul et al. [2023] propose using BERTScore [Zhang et al., 2020], based on the maximum cosine similarity of contextualized word embeddings between token pairs in two candidate texts.

Natural Language Inference (NLI) models are another popular method for evaluating similarity between an original response and candidate responses. These models classify a pair of texts as either *entailment*, *contradiction*, or *neutral*. Several studies propose using NLI estimates of $1 - P(\text{contradiction})$ or $P(\text{entailment})$ between the original response and a set of candidate responses to quantify uncertainty [Chen and Mueller, 2023, Lin et al., 2024]. Zhang et al. [2024] follow a similar approach but instead average across sentences and exclude $P(\text{neutral})$ from their calculations.[5] Other studies compute semantic entropy using NLI-based clustering [Kuhn et al., 2023, Kossen et al., 2024, Farquhar et al., 2024]. Qiu and Miikkulainen [2024] estimate density in semantic space for candidate responses.

White-Box UQ Manakul et al. [2023] consider two scores for quantifying uncertainty with token probabilities: average negative log probability and maximum negative log probability. While these approaches effectively represent a measure of uncertainty, they lack ease of interpretation, are unbounded, and are more useful for ranking than interpreting a standalone score. Fadeeva et al. [2024] consider perplexity, calculated as the exponential of average negative log probability. Similar to average negative log probability, perplexity also has the disadvantage of being unbounded. They also consider response improbability, computed as the complement of the joint token probability of all tokens in the response. Although this metric is bounded and easy to interpret, it penalizes longer token sequences relative to semantically equivalent, shorter token sequences. Another popular metric is entropy, which considers token probabilities over all possible token choices in a pre-defined vocabulary [Malinin and Gales, 2021, Manakul et al., 2023]. Malinin and Gales [2021] also consider the geometric mean of token probabilities for a response, which has the advantage of being bounded and easy to interpret.[6]

LLM-as-a-Judge For uncertainty quantification, several studies concatenate a question-answer pair and ask an LLM to score or classify the answer's correctness. Chen and Mueller [2023] propose using an LLM for self-reflection certainty, where the same LLM is used to judge correctness of the response. Specifically, the LLM is asked to score the response as incorrect, uncertain, or correct, which map to scores of 0, 0.5, and 1, respectively. Similarly, Kadavath et al. [2022] ask the same LLM to state $P(Correct)$ given a question-answer concatenation. Xiong et al. [2024] explore several variations of similar prompting strategies for LLM self-evaluation. More complex variations such as multiple choice question answering generation [Manakul et al., 2023], multi-LLM interaction [Cohen et al., 2023], and follow-up questions [Agrawal et al., 2024] have also been proposed.

[5] Averaging across sentences is done to address long-form responses. Jiang et al. [2024] also address long-form hallucination detection but follow a graph-based approach instead.

[6] For additional white-box uncertainty quantification techniques, we refer the reader to Ling et al. [2024], Bakman et al. [2024], Guerreiro et al. [2023], Zhang et al. [2023], Varshney et al. [2023], Luo et al. [2023], Ren et al. [2023], van der Poel et al. [2022], Wang et al. [2023].

Ensemble Approaches Chen and Mueller [2023] propose a two-component ensemble for zero-resource hallucination known as BSDetector. The first component, known as observed consistency, computes a weighted average of two comparison scores between an original response and a set of candidate responses, one based on exact match, and another based on NLI-estimated contradiction probabilities. The second component is self-reflection certainty, which uses the same LLM to judge correctness of the response. In their ensemble, response-level confidence scores are computed using a weighted average of observed consistency and self-reflection certainty. Verga et al. [2024] propose using a Panel of LLM evaluators (PoLL) to assess LLM responses. Rather than using a single large LLM as a judge, their approach leverages a panel of smaller LLMs. Their experiments find that PoLL outperforms large LLM judges, having less intra-model bias in the judgments.

B Scorer Definitions: Detailed View

B.1 Problem Statement

We aim to model the binary classification problem of whether an LLM response contains a hallucination, which we define as any content that is nonfactual. To this end, we define a collection of binary classifiers, each of which map an LLM response $y_i \in \mathcal{Y}$, generated from prompt x_i, to a 'confidence score' between 0 and 1, where $\mathcal{Y}$ is the set of possible LLM outputs. We denote a hallucination classifier as $\hat{s} : \mathcal{Y} \to [0, 1]$.

Given a classification threshold τ, we denote binary hallucination predictions from the classifier as $\hat{h} : \mathcal{Y} \to \{0, 1\}$. In particular, a hallucination is predicted if the confidence score is less than the threshold τ:

$$\hat{h}(y_i; \theta, \tau) = \mathbb{I}(\hat{s}(y_i; \theta) < \tau), \tag{1}$$

where θ could include additional responses generated from x_i or other parameters. Note that $\hat{h}(\cdot) = 1$ implies a hallucination is predicted. We denote the corresponding ground truth value, indicating whether or not the original response y_i actually contains a hallucination, as $h(y_i)$, where h represents a process to 'grade' LLM responses:

$$h(y_i) = \begin{cases} 1 & y_i \text{ contains a hallucination} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We adapt our scorers from various techniques proposed in the literature. Each scorer outputs response-level confidence scores to be used for hallucination detection. We transform and normalize scorer outputs, if necessary, to ensure each confidence score ranges from 0 to 1 and higher values correspond to greater confidence.[7] Below, we provide details of these various scorers.

B.2 Black-Box UQ Scorers

Black-box UQ scorers exploit variation in LLM responses to the same prompt to assess semantic consistency. For a given prompt x_i, these approaches involve generating m candidate responses $\tilde{\mathbf{y}}_i = \{\tilde{y}_{i1}, ..., \tilde{y}_{im}\}$, using a non-zero temperature, from the same prompt and comparing these responses to the original response y_i. We provide detailed descriptions of each below.

Exact Match Rate. For LLM tasks that have a unique, closed-form answer, exact match rate can be a useful hallucination detection approach. Under this approach, an indicator function is used to score pairwise comparisons between the original response and the candidate responses. Given an original response y_i and candidate responses $\tilde{\mathbf{y}}_i$, generated from prompt x_i, exact match rate (EMR) is computed as follows:

$$EMR(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{I}(y_i = \tilde{y}_{ij}). \tag{3}$$

[7] Note that many of the scorers already have support of $[0, 1]$ and hence do not require normalization.

Non-Contradiction Probability. Non-contradiction probability (NCP) is a similar, but less stringent approach. NCP, a component of the BSDetector approach proposed by Chen and Mueller [2023], also conducts pairwise comparison between the original response and each candidate response. In particular, an NLI model is used to classify each pair $(y_i, \tilde{y}_{ij})$ as *entailment*, *neutral*, or *contradiction* and contradiction probabilities are saved. NCP for original response y_i is computed as the average NLI-based non-contradiction probability across pairings with all candidate responses:

$$NCP(y_i; \tilde{\mathbf{y}}_i) = 1 - \frac{1}{m} \sum_{j=1}^{m} \frac{\eta(y_i, \tilde{y}_{ij}) + \eta(\tilde{y}_{ij}, y_i)}{2} \tag{4}$$

Above, $\eta(y_i, \tilde{y}_{ij})$ denotes the contradiction probability of $(y_i, \tilde{y}_{ij})$ estimated by the NLI model. Following Chen and Mueller [2023] and Farquhar et al. [2024], we use `microsoft/deberta-large-mnli` for our NLI model.

BERTScore. Another approach for measuring text similarity between two texts is BERTScore [Zhang et al., 2020]. Let a tokenized text sequence be denoted as $\mathbf{t} = \{t_1, ...t_L\}$ and the corresponding contextualized word embeddings as $\mathbf{E} = \{\mathbf{e}_1, ..., \mathbf{e}_L\}$, where L is the number of tokens in the text. The BERTScore precision and recall scores between two tokenized texts $\mathbf{t}, \mathbf{t}'$ are respectively defined as follows:

$$BertP(\mathbf{t}, \mathbf{t}') = \frac{1}{|\mathbf{t}|} \sum_{t \in \mathbf{t}} \max_{t' \in \mathbf{t}'} \mathbf{e} \cdot \mathbf{e}'; \quad BertR(\mathbf{t}, \mathbf{t}') = \frac{1}{|\mathbf{t}'|} \sum_{t' \in \mathbf{t}'} \max_{t \in \mathbf{t}} \mathbf{e} \cdot \mathbf{e}' \tag{5}$$

where e, e' respectively correspond to t, t'. We compute our BERTScore confidence (BSC) as follows:

$$BSC(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{m} \sum_{j=1}^{m} 2 \frac{BertP(y_i, \tilde{y}_{ij}) BertR(y_i, \tilde{y}_{ij})}{BertP(y_i, \tilde{y}_{ij}) + BertR(y_i, \tilde{y}_{ij})}, \tag{6}$$

i.e. the average BERTScore F1 score across pairings of the original response with all candidate responses.

Normalized Cosine Similarity. Normalized cosine similarity (NCS) leverages a sentence transformer to map LLM outputs to an embedding space and measure similarity using those sentence embeddings. Let $V : \mathcal{Y} \to \mathbb{R}^d$ denote the sentence transformer, where d is the dimension of the embedding space. We define NCS as the average cosine similarity across pairings of the original response with all candidate responses, normalized by dividing by 2 and adding $\frac{1}{2}$:

$$NCS(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{2m} \sum_{j=1}^{m} \frac{\mathbf{V}(y_i) \cdot \mathbf{V}(\tilde{y}_{ij})}{\|\mathbf{V}(y_i)\| \|\mathbf{V}(\tilde{y}_{ij})\|} + \frac{1}{2}. \tag{7}$$

Normalized Semantic Negentropy. Semantic entropy (SE), proposed by Farquhar et al. [2024], exploits variation in multiple responses to compute a measure of response volatility. The SE approach clusters responses by mutual entailment and, like the NCP scorer, relies on an NLI model. However, in contrast to the aforementioned black-box UQ scorers, semantic entropy does not distinguish between an original response and candidate responses. Instead, it computes a single metric value on a list of responses generated from the same prompt. We consider the discrete version of SE, defined as follows:

$$SE(y_i; \tilde{\mathbf{y}}_i) = -\sum_{C \in \mathcal{C}} P(C|y_i, \tilde{\mathbf{y}}_i) \log P(C|y_i, \tilde{\mathbf{y}}_i), \tag{8}$$

where $P(C|y_i, \tilde{\mathbf{y}}_i)$ denotes the probability a randomly selected response $y \in \{y_i, \tilde{y}_{i1}, ..., \tilde{y}_{im}\}$ belongs to cluster C, and $\mathcal{C}$ denotes the full set of clusters of $\{y_i, \tilde{y}_{i1}, ..., \tilde{y}_{im}\}$.[8] To ensure that we

[8] If token probabilities of the LLM responses are available, the values of $P(C|y_i, \tilde{\mathbf{y}}_i)$ can be instead estimated using mean token probability. However, unlike the discrete case, this version of semantic entropy is unbounded and hence does not lend itself well to normalization.

have a normalized confidence score with $[0, 1]$ support and with higher values corresponding to higher confidence, we implement the following normalization to arrive at *Normalized Semantic Negentropy* (NSN):

$$NSN(y_i; \tilde{\mathbf{y}}_i) = 1 - \frac{SE(y_i; \tilde{\mathbf{y}}_i)}{\log(m+1)}, \tag{9}$$

where $\log(m+1)$ is included to normalize the support.

B.3 White-Box UQ Scorers

White-box UQ scorers leverage token probabilities of the LLM's generated response to quantify uncertainty. We define two white-box UQ scorers below.

Length-Normalized Token Probability. Let the tokenization of LLM response y_i be denoted as $\{t_1, ..., t_{L_i}\}$, where L_i denotes the number of tokens the response. Length-normalized token probability (LNTP) computes a length-normalized analog of joint token probability:

$$LNTP(y_i) = \prod_{t \in y_i} p_t^{\frac{1}{L_i}}, \tag{10}$$

where p_t denotes the token probability for token t.[9] Note that this score is equivalent to the geometric mean of token probabilities for response y_i.

Minimum Token Probability. Minimum token probability (MTP) uses the minimum among token probabilities for a given responses as a confidence score:

$$MTP(y_i) = \min_{t \in y_i} p_t, \tag{11}$$

where t and p_t follow the same definitions as above.

B.4 LLM-as-a-Judge Scorers

We employ LLM-as-a-Judge as an additional method for obtaining response-level confidence scores. LLM-as-a-judge scoring prompts a model to rate the correctness of a question–response concatenation. We adapt the instruction from Xiong et al. [2024] to return a 0–100 score and linearly map it to $[0, 1]$ for consistency with other scorers. Our LLM-as-a-Judge scorer used the following instruction prompt:

> Question: [question], Proposed Answer: [answer].
>
> How likely is the above answer to be correct? Analyze the answer and give your confidence in this answer between 0 (lowest) and 100 (highest), with 100 being certain the answer is correct, and 0 being certain the answer is incorrect. THE CONFIDENCE RATING YOU PROVIDE MUST BE BETWEEN 0 and 100. ONLY RETURN YOUR NUMERICAL SCORE WITH NO SURROUNDING TEXT OR EXPLANATION.
>
> # Example 1
> ## Data to analyze
> Question: Who was the first president of the United States?, Proposed Answer: Benjamin Franklin.
>
> ## Your response
> 4 (highly certain the proposed answer is incorrect)

[9] Although it is not reflected in our notation, the probability for a given token is conditional on the preceding tokens.

Example 2
Data to analyze
Question: What is 2+2?, Proposed Answer: 4

Your response
99 (highly certain the proposed answer is correct)

To ensure a normalized confidence score consistent with the other scorers, we normalize the value returned by the LLM judge to be between 0 and 1. The capitalization and repeated instructions, inspired by Wang et al. [2024], are included to ensure the LLM correctly follows instructions.

B.5 Ensemble Scorer

We introduce a tunable ensemble approach for hallucination detection. Specifically, our ensemble is a weighted average of K binary classifiers: $\hat{s}_k : \mathcal{Y} \to [0, 1]$ for $k = 1, ..., K$. As several of our ensemble components exploit variation in LLM responses to the same prompt, our ensemble is conditional on $(\tilde{\mathbf{y}}_i, \mathbf{w})$, where $\mathbf{w}$ denote the ensemble weights. For original response y_i, we can write our ensemble classifier as follows:

$$\hat{s}(y_i; \tilde{\mathbf{y}}_i, \mathbf{w}) = \sum_{k=1}^{K} w_k \hat{s}_k(y_i; \tilde{\mathbf{y}}_i), \tag{12}$$

where $\mathbf{w} = (w_1, ..., w_K), \sum_{k=1}^{K} w_k = 1$, and $w_k \in [0, 1]$ for $k = 1, ..., K$.[10]

We outline a method for tuning ensemble weights for improved hallucination detection accuracy. This approach allows for customizable component-importance that can be optimized for a specific use case. In practice, tuning the ensemble weights requires having a 'graded' set of n original LLM responses which indicate whether a hallucination is present in each response.[11] For a set of n prompts, we denote the vector of original responses as $\mathbf{y}$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \tag{13}$$

and candidate responses across all prompts with the matrix $\tilde{\mathbf{Y}}$:

$$\tilde{\mathbf{Y}} = \begin{pmatrix} \tilde{\mathbf{y}}_1 \\ \tilde{\mathbf{y}}_2 \\ \vdots \\ \tilde{\mathbf{y}}_n \end{pmatrix} = \begin{pmatrix} \tilde{y}_{11} & \tilde{y}_{12} & \cdots & \tilde{y}_{1m} \\ \tilde{y}_{21} & \tilde{y}_{22} & \cdots & \tilde{y}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{y}_{n1} & \tilde{y}_{n2} & \cdots & \tilde{y}_{nm} \end{pmatrix}. \tag{14}$$

Analogously, we denote the vectors of ensemble confidence scores, binary ensemble hallucination predictions, and corresponding ground truth values respectively as

$$\hat{\mathbf{s}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}) = \begin{pmatrix} \hat{s}(y_1; \tilde{\mathbf{y}}_1, \mathbf{w}) \\ \hat{s}(y_2; \tilde{\mathbf{y}}_2, \mathbf{w}) \\ \vdots \\ \hat{s}(y_n; \tilde{\mathbf{y}}_n, \mathbf{w}) \end{pmatrix}, \tag{15}$$

$$\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}, \tau) = \begin{pmatrix} \hat{h}(y_1; \tilde{\mathbf{y}}_1, \mathbf{w}, \tau) \\ \hat{h}(y_2; \tilde{\mathbf{y}}_2, \mathbf{w}, \tau) \\ \vdots \\ \hat{h}(y_n; \tilde{\mathbf{y}}_n, \mathbf{w}, \tau) \end{pmatrix}, \tag{16}$$

[10] Note that although we write each classifier to be conditional on the set of candidate responses, some of the classifiers depend only on the original response.

[11] Grading responses may be accomplished computationally for certain tasks, e.g. multiple choice questions. However, in many cases, this will require a human grader to manually evaluate the set of responses.

and

$$\mathbf{h}(\mathbf{y}) = \begin{pmatrix} h(y_1) \\ h(y_2) \\ \vdots \\ h(y_n) \end{pmatrix}. \tag{17}$$

Modeling this problem as binary classification enables us to tune the weights of our ensemble classifier using standard classification objective functions. Following this approach, we consider two distinct strategies to tune ensemble weights $w_1, ..., w_K$: threshold-agnostic optimization and threshold-aware optimization.

Threshold-Agnostic Weights Optimization. Our first ensemble tuning strategy uses a threshold-agnostic objective function for tuning the ensemble weights. Given a set of n prompts, corresponding original LLM responses and candidate responses, the optimal set of weights, $\mathbf{w}^*$, is the solution to the following problem:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathcal{W}}{\arg\max}\, \mathcal{S}(\hat{\mathbf{s}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}), \mathbf{h}(\mathbf{y})), \tag{18}$$

where

$$\mathcal{W} = \{(w_1, ..., w_K) : \sum_{k=1}^{K} w_k = 1, w_k \geq 0 \; \forall \; k = 1, ..., K\} \tag{19}$$

is the support of the ensemble weights and $\mathcal{S}$ is a threshold-agnostic classification performance metric, such as area under the receiver-operator characteristic curve (AUROC).

After optimizing the weights, we subsequently tune the threshold using a threshold-dependent objective function. Hence, the optimal threshold, τ^*, is the solution to the following optimization problem:

$$\tau^* = \underset{\tau \in (0,1)}{\arg\max}\, \mathcal{B}(\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}^*, \tau), \mathbf{h}(\mathbf{y})), \tag{20}$$

where $\mathcal{B}$ is a threshold-dependent classification performance metric, such as F1-score.

Threshold-Aware Weights Optimization. Alternatively, practitioners may wish jointly optimize ensemble weights and classification threshold using the same objective. This type of optimization relies on a threshold-dependent objective. We can write this optimization problem as follows:

$$\mathbf{w}^*, \tau^* = \underset{\mathbf{w} \in \mathcal{W}, \tau \in (0,1)}{\arg\max}\, \mathcal{B}(\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}, \tau), \mathbf{h}(\mathbf{y})), \tag{21}$$

where $\mathcal{B}, \hat{\mathbf{h}}$, $\mathbf{h}$, and $\mathcal{W}$ follow the same definitions as above.

C Software Description: `uqlm` Library

The `uqlm` library provides a collection of UQ-based scorers spanning four categories: black-box UQ, white-box UQ, LLM-as-a-Judge, and ensembles. The corresponding classes for these techniques are instantiated by passing an LLM object to the constructor.[12] Each of these classes contains a `generate_and_score` method, which generates LLM responses to a user provided list of prompts and computes response-level confidence scores, which range from 0 to 1.

[12] For the current version of `uqlm`, a LangChain `BaseChatModel` is required. Note that an LLM is not required if users provide pre-generated responses and implement the `score` method.

C.1 Comparison to Existing Toolkits

Traditional grading toolkits (Evals [OpenAI, 2024], G-Eval [Liu et al., 2023]) require ground-truth answers; they are valuable pre-deployment but not usable at generation time. Source-comparison tools (Ragas [Es et al., 2023], Phoenix [Arize AI, 2025], DeepEval [Ip and Vongthongsri, 2025], and others [Hu et al., 2024, UpTrain AI Team, 2024, Zha et al., 2023, Asai et al., 2023]) assess agreement with provided context, yet can validate paraphrases of the prompt without verifying factuality and require sources to be available. Internet-grounded checkers (FacTool [Chern et al., 2023]) introduce latency and potential external errors and do not directly quantify model uncertainty. UQ offerings exist but are narrow or research-oriented: SelfCheckGPT [Manakul et al., 2023] includes a limited subset of scorers and separates generation from evaluation; LangKit [WhyLabs, 2025] and NeMo Guardrails [Rebedea et al., 2023] expose a few UQ signals; LM-Polygraph [Fadeeva et al., 2023] is comprehensive but geared toward research users. `uqlm` bridges these gaps: it provides a zero-resource, generation-time suite across black-box, white-box, and judge signals, standardizes outputs to $[0, 1]$, offers a lightweight, extensible ensemble, and integrates generation and evaluation with a simple API.

C.2 API and Usage

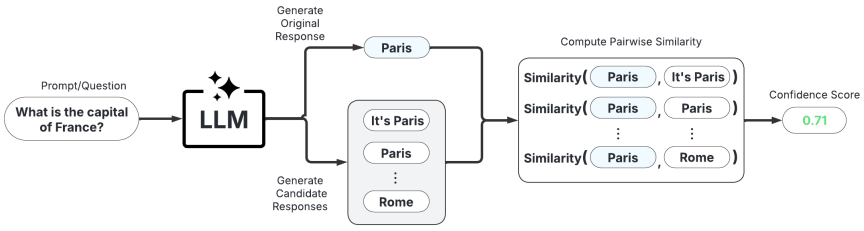


Figure 1: Illustration of a Black-Box Scorer Workflow

C.2.1 Black-Box UQ

Black-box UQ scorers are compatible with any LLM, but increase latency and generation costs. The corresponding class for this collection of scorers is `BlackBoxUQ`. To implement `BlackBoxUQ.generate_and_score`, users provide a list of prompts. For each prompt, an original response, along with additional candidate responses, are generated by the user-provided LLM, and consistency scores are computed using the specified scorers (see Figure 1).[13] If users set `use_best=True`, the uncertainty-minimized response is selected.[14] Below is a minimal example illustrating usage of `BlackBoxUQ`.

```
from uqlm import BlackBoxUQ
bbuq = BlackBoxUQ(llm=llm, scorers=["exact_match", "noncontradiction"])
results = await bbuq.generate_and_score(prompts=prompts, num_responses=5, use_best=True)
```

C.2.2 White-Box UQ

White-box uncertainty quantification leverages token probabilities to compute uncertainty, as depicted in Figure 2. These approaches have the advantage of using the token probabilities associated with the generated response, meaning they do not add any latency or generation cost. However, because token probabilities are not accessible from all APIs, white-box scorers may not be compatible with all LLM applications. This collection of scorers can be implemented with the `WhiteBoxUQ` class. Below is a minimal example of `WhiteBoxUQ` usage.

```
from uqlm import WhiteBoxUQ
wbuq = WhiteBoxUQ(llm=llm, scorers=["min_probability"])
results = await wbuq.generate_and_score(prompts=prompts)
```

[13] Note that Figure 1 depicts the approach for all black-box UQ scorers except semantic entropy, which does not designate an 'original response'.

[14] Uncertainty-minimized response selection is based on semantic entropy [Farquhar et al., 2024].

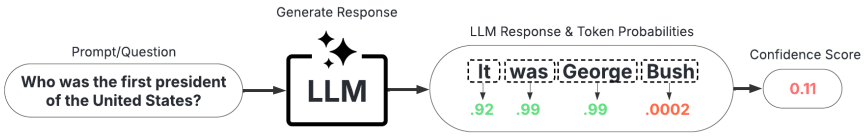


Figure 2: Illustration of a White-Box Scorer Workflow

C.2.3 LLM-as-a-Judge

LLM-as-a-Judge uses an LLM to evaluate the correctness of a response to a particular question. To achieve this, a question-response concatenation is passed to one or more LLMs along with instructions to score the response's correctness using the `LLMPanel` class (see Figure 3). In the constructor, users pass a list of LLM objects to the `judges` argument and specify one of four scoring templates for each judge with the `scoring_templates` argument. These four scoring templates are as follows: binary ($\{\text{incorrect}, \text{correct}\}$ as $\{0, 1\}$), ternary ($\{\text{incorrect}, \text{uncertain}, \text{correct}\}$ as $\{0, 0.5, 1\}$), continuous (any value between 0 and 1), and a 5-point Likert scale $(0, 0.25, ..., 1)$. Implementing the `generate_and_score` method returns the score from each judge and aggregations of these scores, including minimum, maximum, average, and median. See below for a minimal example.

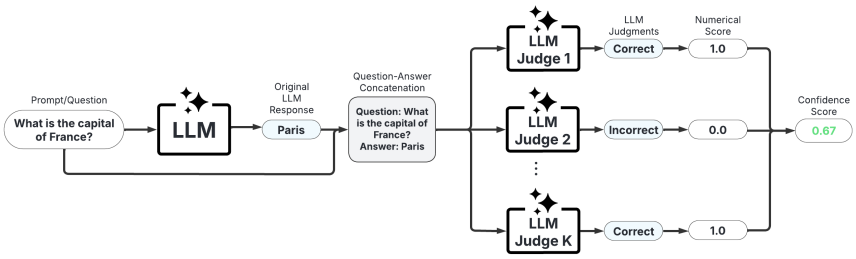


Figure 3: Illustration of LLM-as-a-Judge Workflow

```python
from uqlm import LLMPanel
panel = LLMPanel(llm=llm1, judges=[llm2, llm3], scoring_templates=["continuous", "likert"])
results = await panel.generate_and_score(prompts=prompts)
```

C.2.4 Ensemble Approach

Lastly, `uqlm` offers both tunable and off-the-shelf ensembles that leverage a weighted average of any combination of black-box UQ, white-box UQ, and LLM-as-a-Judge scorers. Similar to the aforementioned classes, `UQEnsemble` enables simultaneous generation and scoring with a `generate_and_score` method. Using the specified scorers, the ensemble score is computed as a weighted average of the individual confidence scores, where weights may be default weights, user-specified, or tuned. If no scorers are specified, the off-the-shelf implementation follows an ensemble of exact match, non-contradiction probability, and self-judge proposed by Chen and Mueller [2023].

In order to tune the ensemble weights prior to using the `generate_and_score` method, users must provide a list of prompts and corresponding ideal responses to serve as an 'answer key'. The LLM's responses to the prompts are graded with a grader function that compares against the provided ideal responses. If a grader function is not provided by the user, the default grader function that leverages `vectara/hallucination_evaluation_model` is used.

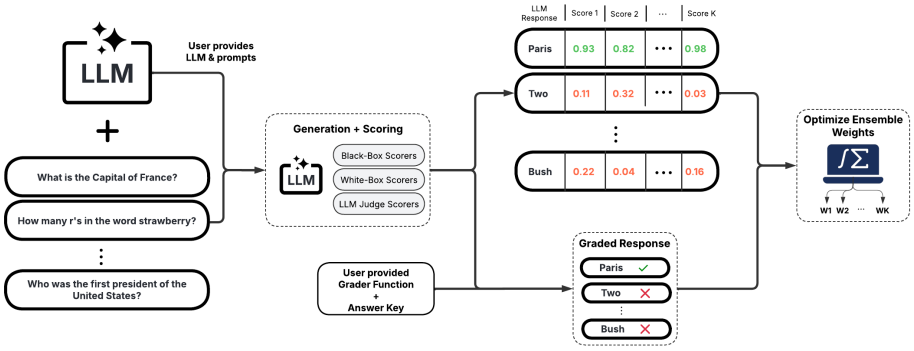


Figure 4: Illustration of Ensemble Tuning

Once the binary grades ('correct' or 'incorrect') are obtained, an optimization routine solves for the optimal weights according to a specified classification objective. The objective function may be threshold-agnostic, such as ROC-AUC, or threshold-dependent, such as F1-score. After completing the optimization routine, the optimized weights are stored as class attributes to be used for subsequent scoring. Below is a minimal example illustrating this process.

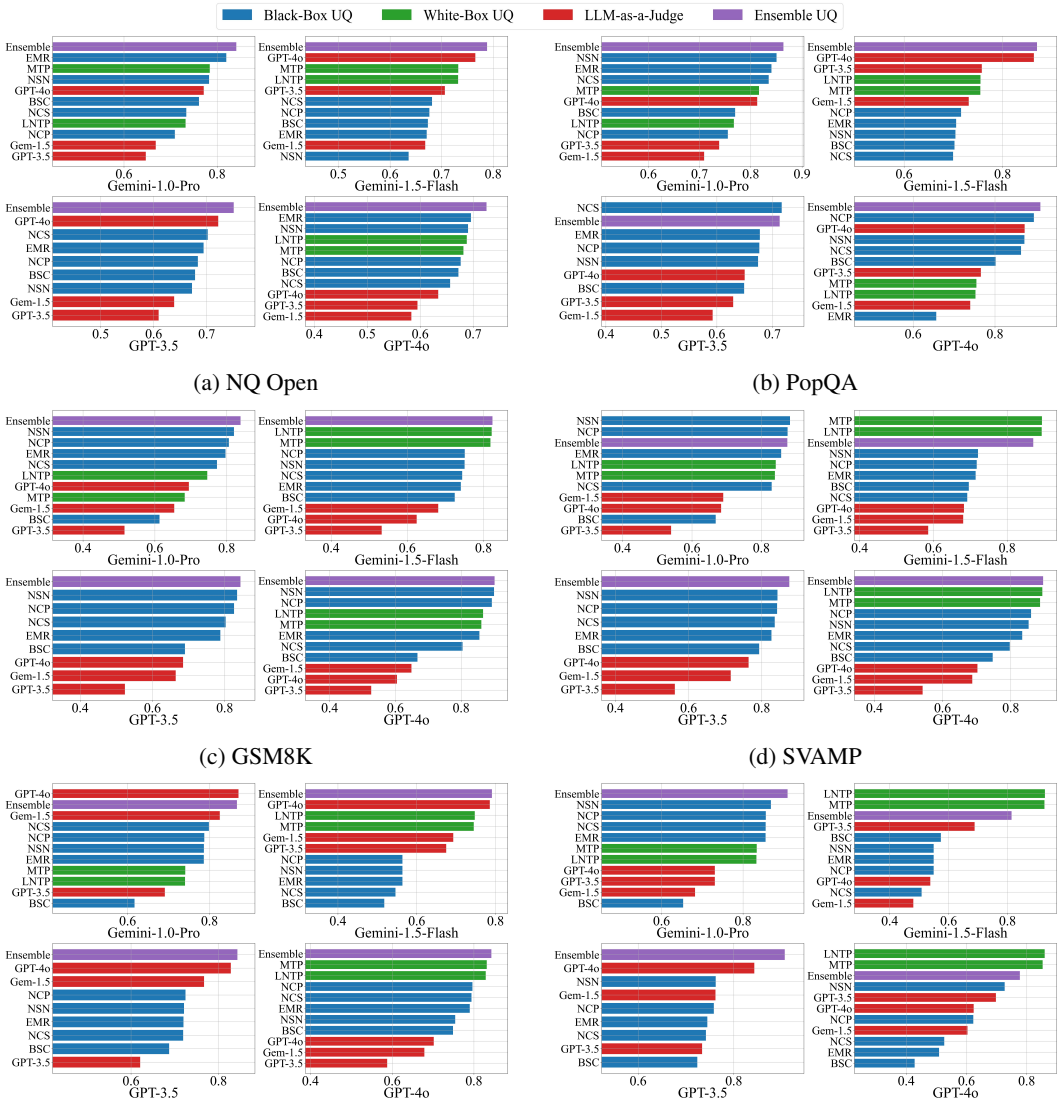
```python
from uqlm import UQEnsemble
## ---Option 1: Off-the-Shelf Ensemble (Chen & Mueller, 2023)---
# uqe = UQEnsemble(llm=llm)
# results = await uqe.generate_and_score(prompts=prompts, num_responses=5)

## ---Option 2: Tuned Ensemble---
scorers = [ # specify which scorers to include
    "exact_match", "noncontradiction", # black-box scorers
    "min_probability", # white-box scorer
    llm # use same LLM as a judge
]
uqe = UQEnsemble(llm=llm, scorers=scorers)

# Tune on tuning prompts with provided ground truth answers
tune_results = await uqe.tune(
    prompts=tuning_prompts, ground_truth_answers=ground_truth_answers
)
# ensemble is now tuned - generate responses on new prompts
results = await uqe.generate_and_score(prompts=prompts)
results.to_df()
```

D Additional Figures from Experiments

Below we present additional figures and tables from our experiments: (i) scorer-specific AUROC, (ii) scorer-specific F1, (iii) filtered accuracy vs confidence threshold for the top scorer from each family, (iv) AUROC vs number of sampled candidates for all black-box scorers, and (v) Average exact match rate. Each figure and table covers all 24 LLM–dataset scenarios.



(a) NQ Open

(b) PopQA

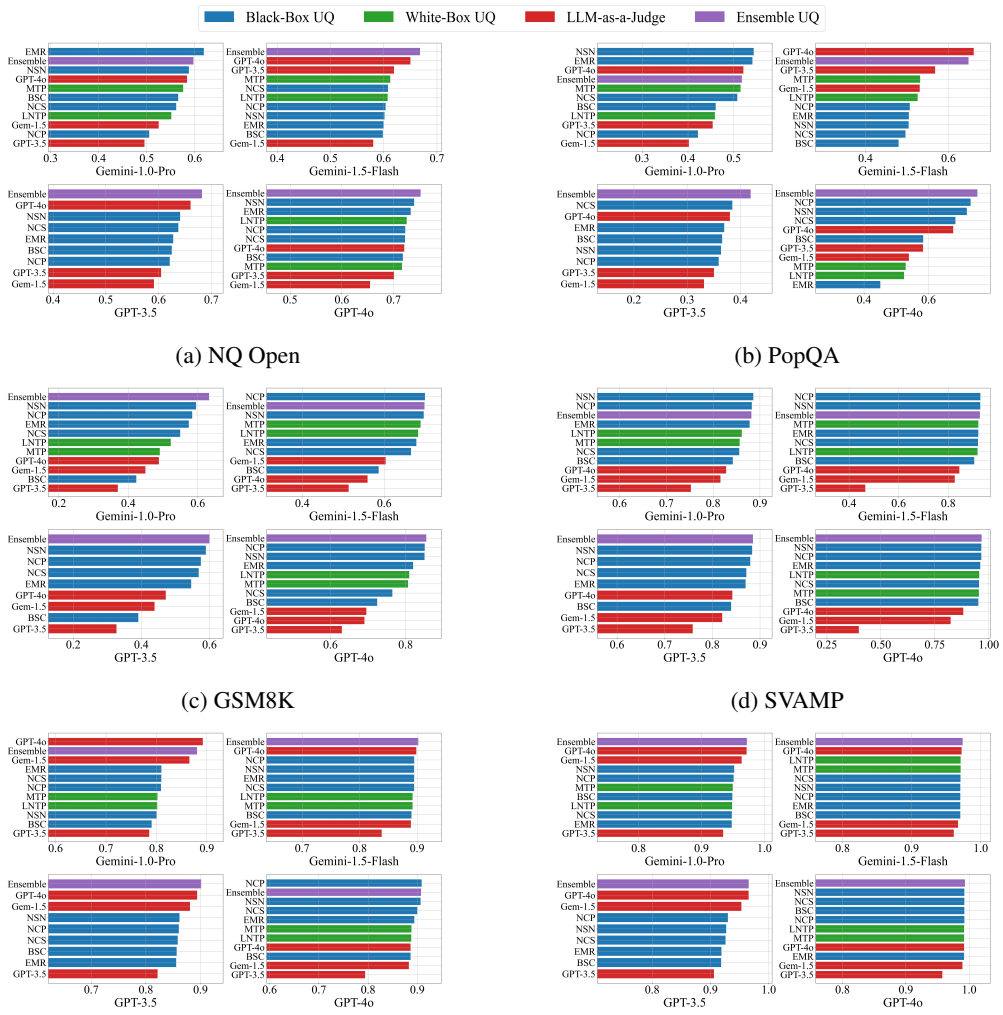
(c) GSM8K

(d) SVAMP

(e) CSQA

(f) AI2-ARC

Figure 5: Scorer-Specific AUROC Scores for Hallucination Detection by LLM and Dataset (Higher is Better)



(a) NQ Open

(b) PopQA

(c) GSM8K

(d) SVAMP

(e) CSQA

(f) AI2-ARC

Figure 6: Scorer-Specific F1-Scores for Hallucination Detection by LLM and Dataset (Higher is Better)

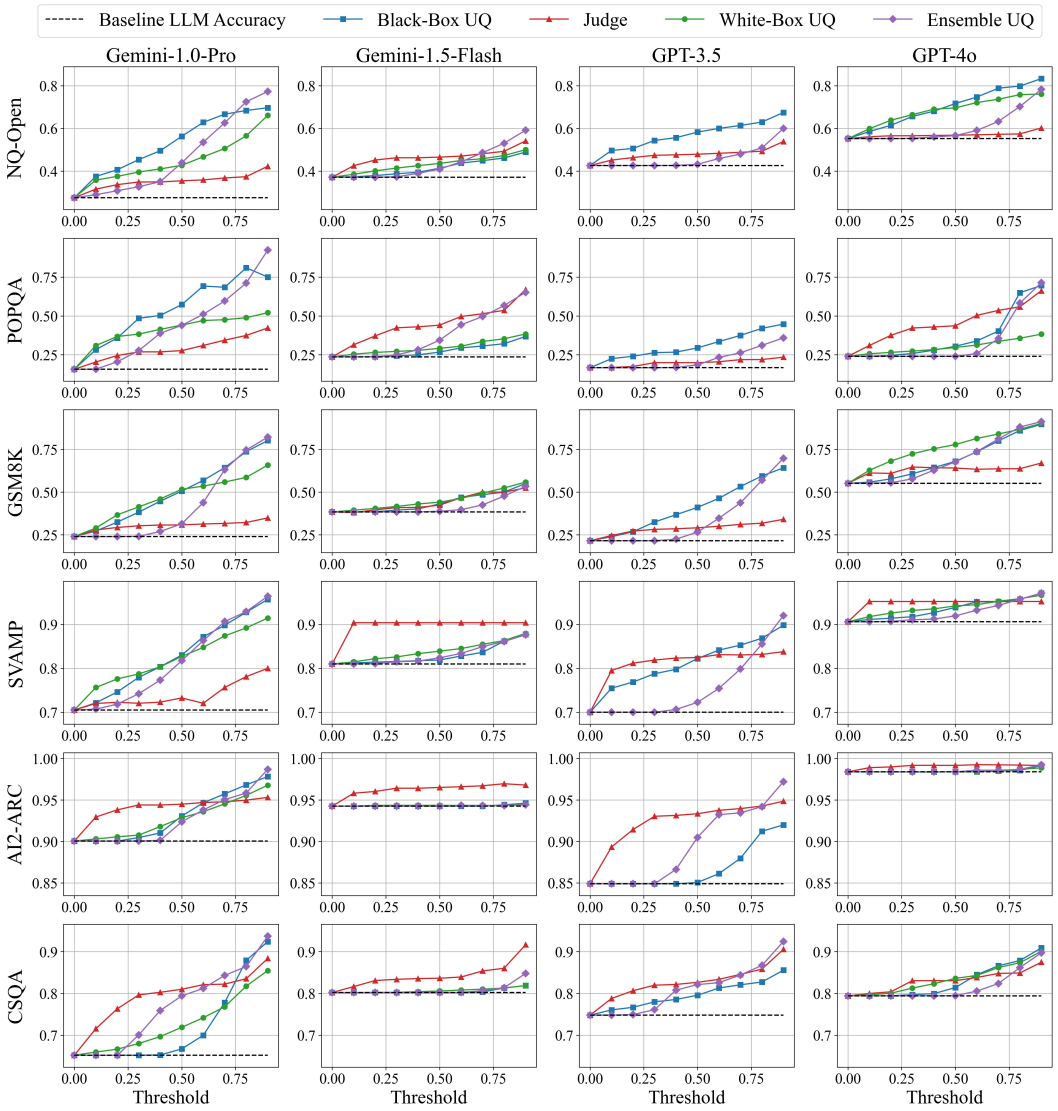


Figure 7: Filtered LLM Accuracy vs. Confidence Threshold (Top per Scorer Type)

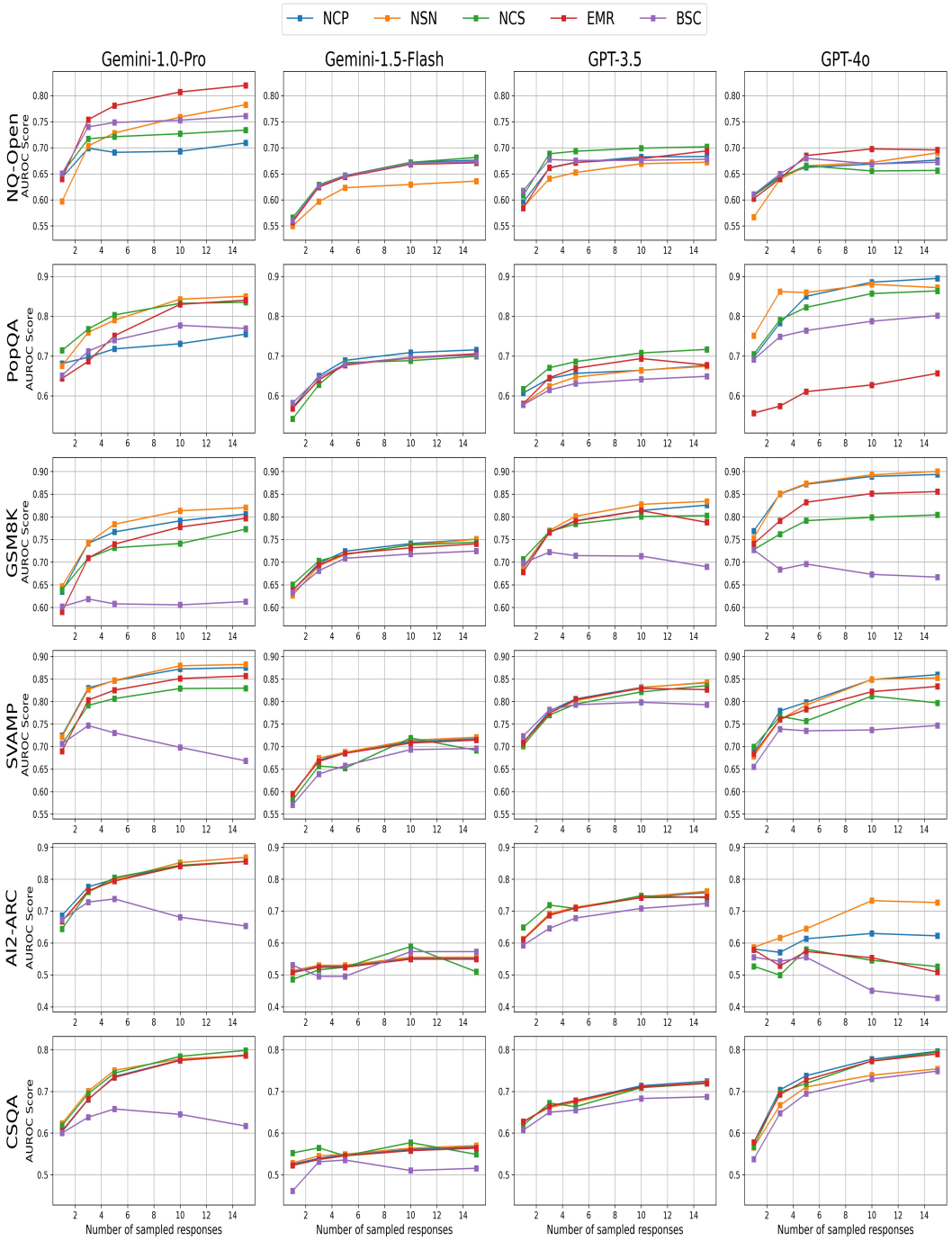


Figure 8: Hallucination Detection AUROC by Number of Sampled Responses

Table 4: Average Exact Match Rate by LLM and Dataset

Model Used	NQ-Open	PopQA	GSM8K	SVAMP	CSQA	AI2-ARC
Gemini-1.5-Flash	0.81	0.79	0.83	0.96	0.99	1.00
Gemini-1.0-Pro	0.36	0.18	0.25	0.66	0.71	0.85
GPT-3.5	0.47	0.29	0.30	0.76	0.81	0.89
GPT-4o	0.44	0.35	0.63	0.91	0.90	0.81