Uncertainty Quantification for Language Models: A Suite of Black-Box, White-Box, LLM Judge, and Ensemble Scorers

Anonymous authors
Paper under double-blind review

Abstract

Hallucinations are a persistent problem with Large Language Models (LLMs). As these models become increasingly used in high-stakes domains, such as healthcare and finance, the need for effective hallucination detection is crucial. To this end, we outline a versatile framework for zero-resource hallucination detection that practitioners can apply to real-world use cases. To achieve this, we adapt a variety of existing uncertainty quantification (UQ) techniques, including black-box UQ, white-box UQ, and LLM-as-a-Judge, transforming them as necessary into standardized response-level confidence scores ranging from 0 to 1. To enhance flexibility, we propose a tunable ensemble approach that incorporates any combination of the individual confidence scores. This approach enables practitioners to optimize the ensemble for a specific use case for improved performance. To streamline implementation, the full suite of scorers is offered in this paper's companion Python toolkit. To evaluate the performance of the various scorers, we conduct an extensive set of experiments using several LLM question-answering benchmarks. We find that our tunable ensemble typically surpasses its individual components and outperforms existing hallucination detection methods. Our results demonstrate the benefits of customized hallucination detection strategies for improving the accuracy and reliability of LLMs.

1 Introduction

Large language models (LLMs) are being increasingly used in production-level applications, often in high-stakes domains such as healthcare or finance. Consequently, there is a rising need to monitor these systems for the accuracy and factual correctness of model outputs. In these sensitive use cases, even minor errors can pose serious safety risks while also leading to high financial costs and reputational damage. A particularly concerning risk for LLMs is hallucination, where LLM outputs sound plausible but contain content that is factually incorrect. Many studies have investigated hallucination risk for LLMs (see Huang et al. (2023); Tonmoy et al. (2024); Shorinwa et al. (2024); Huang et al. (2024) for surveys of the literature). Even recent models, such as OpenAI's GPT-4.5, have been found to hallucinate as often as 37.1% on certain benchmarks (OpenAI, 2025), underscoring the ongoing challenge of ensuring reliability in LLM outputs.

Hallucination detection methods typically involve comparing ground truth texts to generated content, comparing source content to generated content, or quantifying uncertainty. Assessments that compare ground truth texts to generated content are typically conducted pre-deployment in order to quantify hallucination risk of an LLM for a particular use case. While important, this collection of techniques does not lend itself well to real-time evaluation and monitoring of systems already deployed to production. In contrast, techniques that compare source content to generated content or quantify uncertainty can compute response-level scores at generation time and hence can be used for real-time monitoring of production-level applications.

Uncertainty quantification (UQ) techniques can be used for hallucination detection in a zero-resource fashion, meaning they do not require access to a database of source content, ground truth texts, or internet access. These approaches are typically classified as either black-box UQ, white-box UQ, or LLM-as-a-Judge. Black-box UQ methods exploit the stochastic nature of LLMs and measure semantic consistency of multiple responses generated from the same prompt. White-box UQ methods leverage token probabilities associated

with the LLM outputs to compute uncertainty or confidence scores. LLM-as-a-Judge methods use one or more LLMs to evaluate the factual correctness of a question-answer concatenation.

In this paper, we outline a versatile framework for generation-time, zero-resource hallucination detection that practitioners can apply to real-world use cases. To achieve this, we adapt a variety of existing black-box UQ, white-box UQ, and LLM-as-a-Judge methods, applying transformations as necessary to obtain standardized confidence scores that range from 0 to 1. For improved customization, we propose a tunable ensemble approach that incorporates any combination of the individual scorers. The ensemble output is a simple weighted average of these individual components, where the weights can be tuned using a user-provided set of graded LLM responses. This approach enables practitioners to optimize the ensemble for a specific use case, leading to more accurate and reliable hallucination detection. Importantly, our ensemble is extensible, meaning practitioners can expand to include new components as research on hallucination detection evolves.

We evaluate the full suite of UQ scorers on responses generated by four LLMs across various question-answer benchmarks, yielding several empirical insights. Most notably, our tunable ensemble generally outperforms its individual components for hallucination detection. However, the performance ranking of individual scorers varies by dataset, underscoring the value of tailoring methods to specific use cases. Among black-box UQ scorers, entailment-style approaches demonstrate superior performance in our comparisons, while all black-box methods show reduced effectiveness when variation in sampled responses is low. Furthermore, gains from sampling additional responses shrink as the number of candidate responses rises, providing practical guidance for deployment. Lastly, a model's accuracy on a given dataset positively relates to its performance as a judge of other models' answers on that dataset.

Finally, this paper is complemented by our open-source Python package that provides ready-to-use implementations of all uncertainty quantification methods presented and evaluated in this work.[1] This library enables practitioners to generate responses and obtain response-level confidence scores by providing prompts (i.e., the questions or tasks for the LLM) along with their chosen LLM. Our framework and toolkit provide researchers and developers a model-agnostic, user-friendly way to implement our suite of UQ-based scorers in real-world use cases, enabling more informed decisions around LLM outputs.

2 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 & 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 & 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 & Mueller,

[1] To maintain anonymity, the library's anonymized source code is provided in this submission's supplemental material. The link to the open-source repository will be provided upon acceptance.

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.[2] Other studies compute semantic entropy using NLI-based clustering (Kuhn et al., 2023; Kossen et al., 2024; Farquhar et al., 2024). Qiu & 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 & Gales, 2021; Manakul et al., 2023). Malinin & Gales (2021) also consider the geometric mean of token probabilities for a response, which has the advantage of being bounded and easy to interpret.[3]

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 & 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.

Ensemble Approaches Chen & 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.

3 Hallucination Detection Methodology

3.1 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 $\hat{s} : \mathcal{Y} \rightarrow [0, 1]$.

[2]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.

[3]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).

Given a classification threshold τ, we denote binary hallucination predictions from the classifier as $\hat{h} : \mathcal{Y} \rightarrow \{0, 1\}$. 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 the prompt x_i, 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 label indicating whether the original response y_i actually contains a hallucination as $h(y_i; \cdot)$. The function h represents a grading process that compares a generated response y_i to a correct reference answer y_i^* for the same prompt x_i:

$$h(y_i; y_i^*, x_i) = \begin{cases} 1 & y_i \text{ is incorrect with respect to } y_i^* \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

When the correct answer y_i^* is known, direct comparison yields the most accurate hallucination label. In deployment, however, y_i^* is not available at generation time, so h cannot be used as a real-time classifier. We therefore treat h as an oracle ground truth used only offline for evaluation and, where applicable, tuning. Our framework approximates h with $\hat{h}$ using uncertainty-based signals available at generation time, consistent with our zero-resource setting and without access to y_i^*.

We adapt our uncertainty-based 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.[4] Below, we provide details of these various scorers.

3.2 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:

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

Non-Contradiction Probability. Non-contradiction probability (NCP) is a similar, but less stringent approach. NCP, a component of the BSDetector approach proposed by Chen & 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:

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

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

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

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:

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

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 $\frac{1}{2}$:

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

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:

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

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 $\{y_i, \tilde{y}_{i1}, ..., \tilde{y}_{im}\}$.[5] To ensure that we 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):

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

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

3.3 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.

[5] 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.

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:

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

where p_t denotes the token probability for token t.[6] 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:

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

where t and p_t follow the same definitions as above.

3.4 LLM-as-a-Judge Scorers

We employ an LLM-as-a-Judge scorer $J(y_i; x_i)$ as an additional method for obtaining response-level confidence scores.[7] 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 A.

3.5 Ensemble Scorer

We introduce a tunable ensemble approach for hallucination detection. Specifically, our ensemble is a weighted average of K binary classifiers: $\hat{s}_k : \mathcal{Y} \rightarrow [0, 1]$ 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:

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

where $\mathbf{w} = (w_1, ..., w_K), \sum_{k=1}^{K} w_k = 1$, and $w_k \in [0, 1]$ for $k = 1, ..., K$.[8]

Tuning the ensemble requires a sample of LLM responses $y_1, ..., y_n$ to a set of n prompts, a set of K confidence scores for each response $\{s_1(y_i; \tilde{\mathbf{y}}_i, x_i), ..., s_K(y_i; \tilde{\mathbf{y}}_i, x_i)\}_{i=1}^{N}$, and corresponding binary hallucination indicators $h(y_1; y_1^*, x_1), ..., h(y_n; y_n^*, x_n)$.[9] Given a classification objective function, the ensemble weights $\mathbf{w}$ can be tuned with an optimization routine.[10] If the objective is threshold-agnostic, the weights and threshold τ can be tuned sequentially. For a threshold-dependent objective, the weights and threshold can be tuned jointly. See Appendix B for more details on ensemble tuning.

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

[7] Our use of "zero-resource" means no external corpora, retrieval, ground-truth texts, or internet search access at scoring time. An LLM judge satisfies this because it operates only on (x_i, y_i) and the instruction prompt, using its fixed internal parameters. We do not enable tools, browsing, code execution, or retrieval, nor do we supply auxiliary context beyond the question and response. A tool- or retrieval-augmented judge would fall outside our zero-resource scope. While we acknowledge that the judge's internal knowledge can influence judgments, we do not consider this an external resource requirement.

[8] 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.

[9] Note that $h(y_1; y_1^*, x_1), ..., h(y_n; y_n^*, x_n)$ serve as 'ground truth' labels in the classification objective function.

[10] We use `optuna` (Akiba et al., 2019) for optimization with default settings (more details available here).

4 Experiments

4.1 Experiment Setup

We conduct a series of experiments to assess the hallucination detection performance of the various scorers. To accomplish this, we leverage a set of publicly available benchmark datasets that contain questions and answers. To ensure that our answer format has sufficient variation, we use two benchmarks with numerical answers (*GSM8K* (Cobbe et al., 2021) and *SVAMP* (Patel et al., 2021)), two with multiple-choice answers (*CSQA* (Talmor et al., 2022) and *AI2-ARC* (Clark et al., 2018)), and two with open-ended text answers (*PopQA* (Mallen et al., 2023) and *NQ-Open* (Lee et al., 2019)).

We sample 1000 questions for each of the six benchmarks. For each question, we generate an original response and $m = 15$ candidate responses using four LLMs: GPT-3.5 (OpenAI), GPT-4o (OpenAI), Gemini-1.0-Pro (Google), and Gemini-1.5-Flash (Google).[11] Candidate responses are generated using a temperature of 1.0. For each response, we use the corresponding candidate responses generated by the same LLM to compute the full suite of black-box UQ scores. We also compute LLM-as-a-Judge scores for each response using three different judge models: GPT-3.5, GPT-4o, and Gemini-1.5-Flash.[12] Lastly, we compute white-box scores for the GPT-4o, Gemini-1.0-Pro, and Gemini-1.5-Flash responses.[13] We evaluate the hallucination detection performance of all individual scorers as well as our ensemble scorer for each of the benchmarks using various metrics, with grading processes detailed in Appendix C. All scores are computed using this paper's companion toolkit.[14]

4.2 Experiment Results

Threshold-Agnostic Evaluation To assess the performance of the scorers as hallucination classifiers, we evaluate the performance of the confidence scores in a threshold-agnostic fashion.[15] Under this setting, we use the AUROC-optimized ensemble weights and compute the ensemble's AUROC using 5-fold cross-validation. The final reported AUROC is obtained by averaging the AUROC values across the five holdout sets.

Table 1: Hallucination Detection AUROC (Higher is Better): Best-Performing Scorer by LLM and Dataset

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

Figure 1 presents the AUROC scores for all scorers across the 24 LLM-dataset scenarios, while Table 1 highlights the best-performing scorer for each scenario. The AUROC values for the scenario-specific best scorers range from 0.72 for GPT-3.5 responses on PopQA (NCS) to 0.93 for Gemini-1.5-Flash responses on AI2-ARC (LNTP). Overall, the top-performing scorers for each scenario exhibit strong hallucination

[11] We use a large number of candidate responses ($m = 15$) to ensure robust comparisons across black-box scorers. In practice, fewer candidates can be used. For an experimental evaluation of the impact of m on performance, refer to Appendix D.

[12] We were unable to use Gemini-1.0-Pro as a judge model because it was retired during the course of our experiments.

[13] Our instance of GPT-3.5 did not support token probability access. Hence, we were unable to compute white-box UQ scores for the GPT-3.5 responses.

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

[15] In our experiments, we label 'correct' LLM responses as 1 and 'incorrect' responses as 0.

detection performance, with AUROC values greater than 0.8 for 19 out of 24 scenarios. However, some scorers perform only slightly better than a random classifier in hallucination detection in certain scenarios, such as the GPT-3.5 LLM judge for all four GSM8K scenarios.

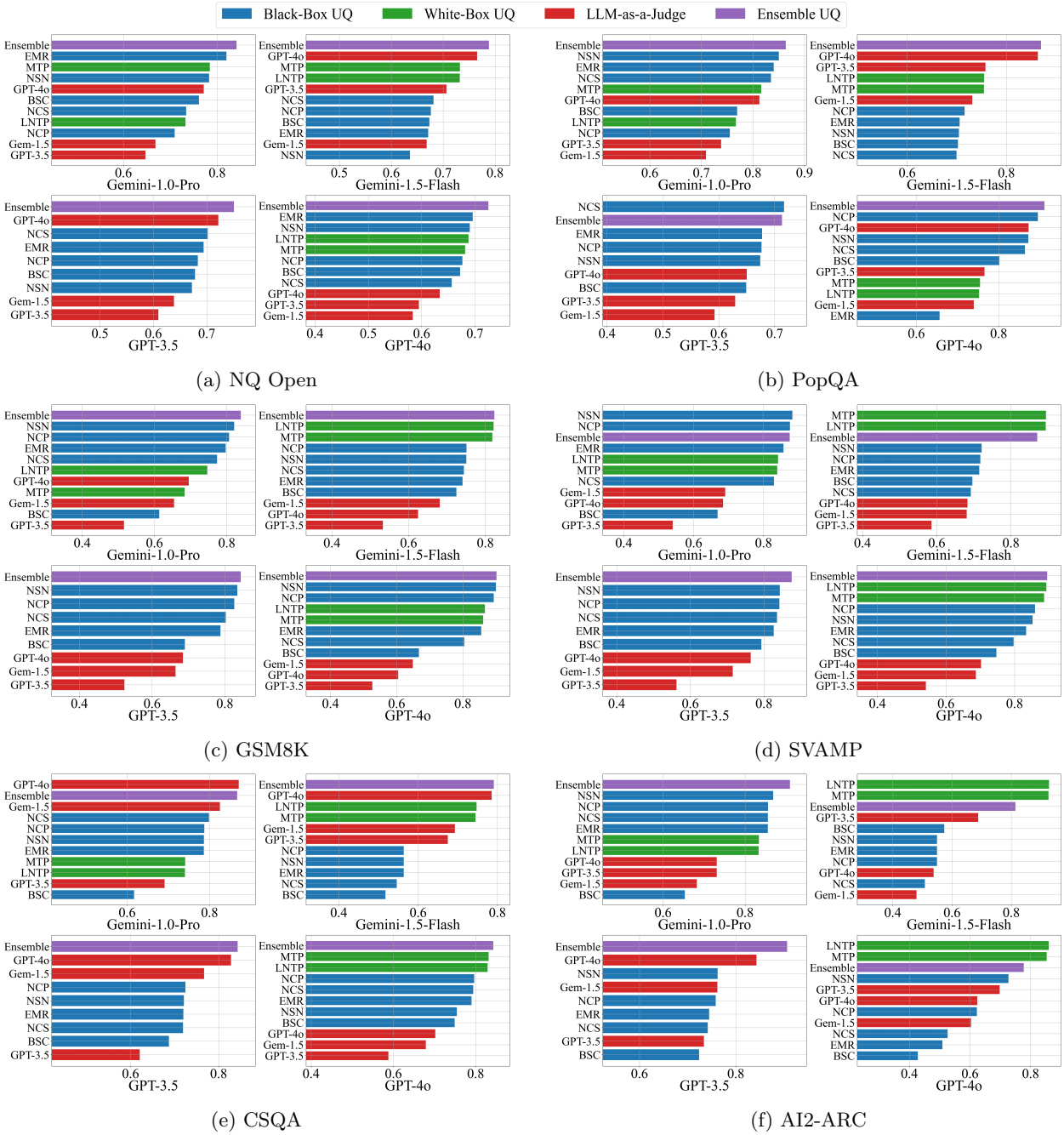

Figure 1: Scorer-Specific AUROC Scores for Hallucination Detection by LLM and Dataset (Higher is Better)

When comparing AUROC values across scorers, we find our ensemble scorer outperforms its individual components in 18 out of 24 scenarios, demonstrating the benefits of use-case-specific optimization. The rankings of individual scorers, however, vary significantly across scenarios, with black-box, LLM-as-a-Judge, and white-box methods achieving top non-ensemble performance in 11, 8, and 5 scenarios respectively. Consistent with previous studies (Kuhn et al., 2023; Manakul et al., 2023; Lin et al., 2024; Farquhar et al.,

2024), NLI-based scorers (NSN and NCP) typically lead performance among black-box scorers, achieving the highest AUROC in 18 out of 24 scenarios. Among LLM judges, GPT-4o consistently outperforms GPT-3.5 and Gemini-1.5-Flash, ranking highest in 19 out of 24 scenarios. Finally, the two white-box scorers perform approximately equally, with very similar AUROC scores in the vast majority of scenarios.

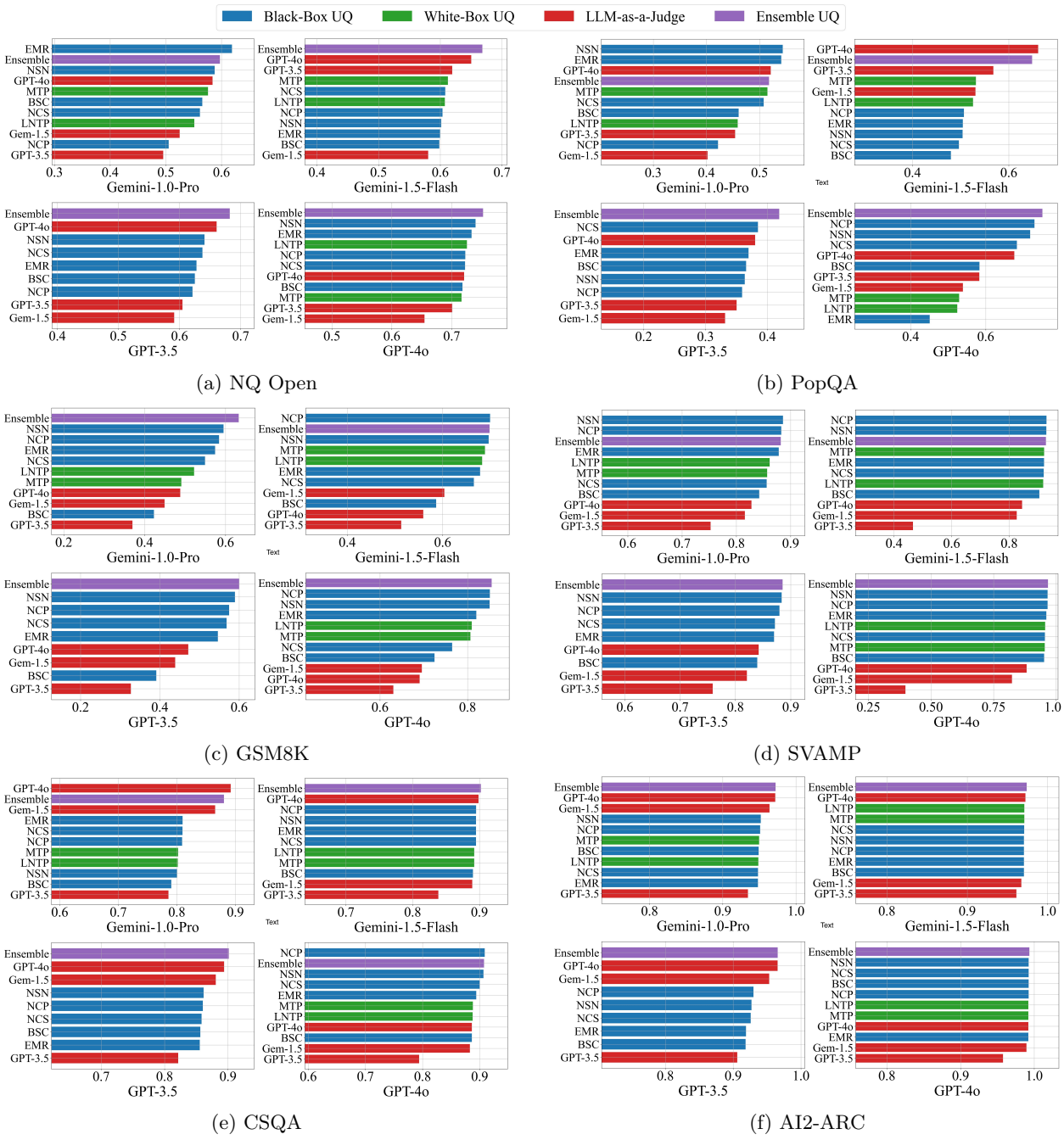

Figure 2: Scorer-Specific F1-Scores for Hallucination Detection by LLM and Dataset (Higher is Better)

Threshold-Optimized Evaluation. We evaluate the various scorers using a threshold-dependent metric (F1-score). To compute our ensemble scores in this setting, we jointly optimize the ensemble weights and threshold using F1-score as the objective function, as outlined in Appendix B. To ensure robust evaluations, we compute the scorer-specific F1-scores using 5-fold cross-validation. For each individual scorer, we select

the F1-optimal threshold using grid search on the tuning set and compute F1-score on the holdout set. We report the final F1-score for each scorer as the average across holdout sets.

Figure 2 displays the F1-scores for each scorer, while Table 2 summarizes the precision, recall, and F1 metrics for the top-performing scorer across all 24 evaluation scenarios. Overall, the results are consistent with the AUROC experiments. Again, the ensemble scorer outperforms its individual components in most scenarios, achieving highest F1-score in 16 out of 24 scenarios. As in the AUROC experiments, the NLI-based scorers (NSN and NCP) are consistently the top-performing black-box scorers (19 out of 24 scenarios), GPT-4o is consistently the top-performing judge (22 out of 24 scenarios), and the two white-box scorers perform approximately equally. Interestingly, the strong performance of GPT-4o as an LLM judge is consistent with its performance in answering questions as the original LLM, where it outperforms GPT-3.5, Gemini-1.5-Flash, and Gemini-1.0-Pro in baseline accuracy across all six benchmarks (see Figure 3).

Table 2: Hallucination Detection F1-Scores (Higher is Better): Best-Performing Scorer by LLM and Dataset

Model	Metric	NQ-Open	PopQA	GSM8K	SVAMP	CSQA	AI2-ARC
Gem.-1.0-Pro	Precision	0.54	0.45	0.58	0.86	0.82	0.95
	Recall	0.73	0.69	0.70	0.91	0.98	1.00
	F1-Score	0.62	0.54	0.63	0.89	0.89	0.97
	Best Scorer	EMR	NSN	Ensemble	NSN	GPT-4o	Ensemble
Gem.-1.5-Flash	Precision	0.62	0.55	0.58	0.89	0.84	0.95
	Recall	0.75	0.82	0.88	0.98	0.97	1.00
	F1-Score	0.67	0.66	0.70	0.93	0.90	0.97
	Best Scorer	Ensemble	GPT-4o	NCP	NCP	Ensemble	Ensemble
GPT-3.5	Precision	0.57	0.32	0.54	0.83	0.85	0.94
	Recall	0.85	0.79	0.71	0.94	0.96	1.00
	F1-Score	0.68	0.42	0.60	0.89	0.90	0.97
	Best Scorer	Ensemble	Ensemble	Ensemble	Ensemble	Ensemble	Ensemble
GPT-4o	Precision	0.67	0.72	0.84	0.95	0.85	0.99
	Recall	0.87	0.79	0.88	0.99	0.97	1.00
	F1-Score	0.75	0.75	0.85	0.97	0.91	0.99
	Best Scorer	Ensemble	Ensemble	Ensemble	Ensemble	NCP	Ensemble

Filtered Accuracy@τ. Lastly, we compute model accuracy on the subset of LLM responses having confidence scores exceeding a specified threshold τ. We refer to this metric as *Filtered Accuracy@τ*. Since the LLM accuracy depends on the choice of the threshold τ, we repeat the calculation for $\tau = 0, 0.1, ..., 0.9$. Note that accuracy at $\tau = 0$ uses the full sample without score-based filtering.

Figure 3 presents the Filtered Accuracy@τ for the highest performing white-box, black-box, LLM-as-a-Judge, and ensemble scorers. Across all scenarios, response filtering with the highest-performing scorers leads to an approximately monotonic increase in LLM accuracy as the threshold increases. For example, when filtering Gemini-1.0-Pro responses to PopQA questions using the leading black-box scorer, accuracy improves dramatically from a baseline of 0.15 to 0.69 at $\tau = 0.6$. Similarly, with GPT-4o responses on GSM8K, filtering with the white-box scorer achieves 0.81 accuracy at $\tau = 0.6$, substantially higher than the baseline accuracy of 0.55.

A notable exception to this trend occurs with black-box scorers on Gemini-1.5-Flash responses to CSQA and AI2-ARC datasets, where no accuracy gains are observed despite filtering. This absence of improvement is consistent with the remarkably low AUROC values for these scorers in the same scenarios shown in Figure 1. Our supplemental analysis in Appendix D reveals that this limitation stems from the lack of response diversity in these two scenarios, with average exact match rates of 0.99 and 1.00, respectively, indicating near-uniformity in candidate responses that provide little signal for black-box scorers to leverage.

5 Discussion

Choosing Among Scorers. Choosing the right confidence scorer for an LLM system depends on several factors, including API support, latency requirements, LLM behavior, and the availability of graded datasets. If the API supports access to token probabilities in LLM generations, white-box scorers can be implemented without adding latency or generation costs. However, if the API does not provide access to token probabilities, black-box scorers and LLM-as-a-Judge may be the only feasible options. When choosing among black-box and LLM-as-a-Judge scorers, latency requirements are a key consideration. For low-latency applications, practitioners should avoid higher-latency black-box scorers such as NLI-based scorers (NSN and NCP), opting instead for faster black-box scorers or LLM-as-a-Judge. If latency is not a concern, any of the black-box scorers may be suitable.

Black-box scorers may struggle when LLMs exhibit minimal variation in sampled responses, as shown in our experiments (Appendix D). In such cases, white-box scorers or LLM-as-a-Judge approaches are preferable. For LLM-as-a-Judge implementations, our findings reveal that an LLM's accuracy on a specific dataset positively relates to its ability to judge responses to questions from that same dataset, providing a practical criterion for judge selection. Finally, if a graded dataset is available, practitioners can tune an ensemble of various confidence scores to improve hallucination detection, as detailed in Appendix B. Our experiments demonstrate that a tuned ensemble can potentially provide more accurate confidence scores than individual scorers. By considering these factors and choosing the right confidence score, practitioners can improve the performance of their LLM system.

Using Confidence Scores. In practice, practitioners may wish to use confidence scores for various purposes. First, practitioners can use our confidence scores for response filtering, where responses with low confidence are blocked, or 'targeted' human-in-the-loop, where low-confidence responses are selected for manual review. Our experimental evaluations of Filtered Accuracy@τ demonstrate the efficacy of these approaches, illustrating notable improvements in LLM accuracy when low-confidence responses are filtered out. Note that selecting a confidence threshold for flagging or blocking responses will depend on the scorer used, the dataset being evaluated, and stakeholder values (e.g., relative cost of false negatives vs. false positives).

Alternatively, confidence scores can be used for pre-deployment diagnostics, providing practitioners insights into the types of questions on which their LLM is performing worst. Findings from this type of exploratory analysis can inform strategies for improvements, such as further prompt engineering. Overall, the scorers included in our framework and toolkit provide practitioners with an actionable way to improve response quality, optimize resource allocation, and mitigate risks.

We note two important ethical considerations related to using confidence scores. First, a confidence score reflects model uncertainty rather than ground-truth-based correctness, and high scores can induce over-reliance in high-risk settings. We therefore caution against using confidence scores as decision guarantees in domains such as medicine, law, or finance without external human review and domain-specific safeguards. Second, we encourage monitoring for distributional fairness and content diversity, including per-segment error-rate evaluation, and, where applicable, safeguards that prevent systematic removal of accurate responses for particular groups.

Limitations and Future Work. We note a few important limitations of this work. First, although our experiments leverage six question-answering benchmark datasets spanning three types of questions, we acknowledge that our findings on scorer-specific performance may not generalize to other types of questions.[16] For example, long-form generation (such as summarization) can mix true and false claims within a single output and may require claim-level decomposition to obtain useful uncertainty signals. Likewise, correctness in code generation depends on syntax and execution, so black-box consistency measurements will likely require alternative methods.

Second, while we conduct experiments using four LLMs, performance of the various scorers may differ for other LLMs. Note that for different LLMs, differences in token probability distributions will impact the behavior of white-box scorers, and the degree of variation in responses to the same prompt will affect

[16] For practical reasons, we selected benchmark datasets containing questions that could be graded computationally.

the performance of black-box scorers. LLM-as-a-Judge may have better performance using more recently released, higher-performance LLMs or alternative instruction prompts.

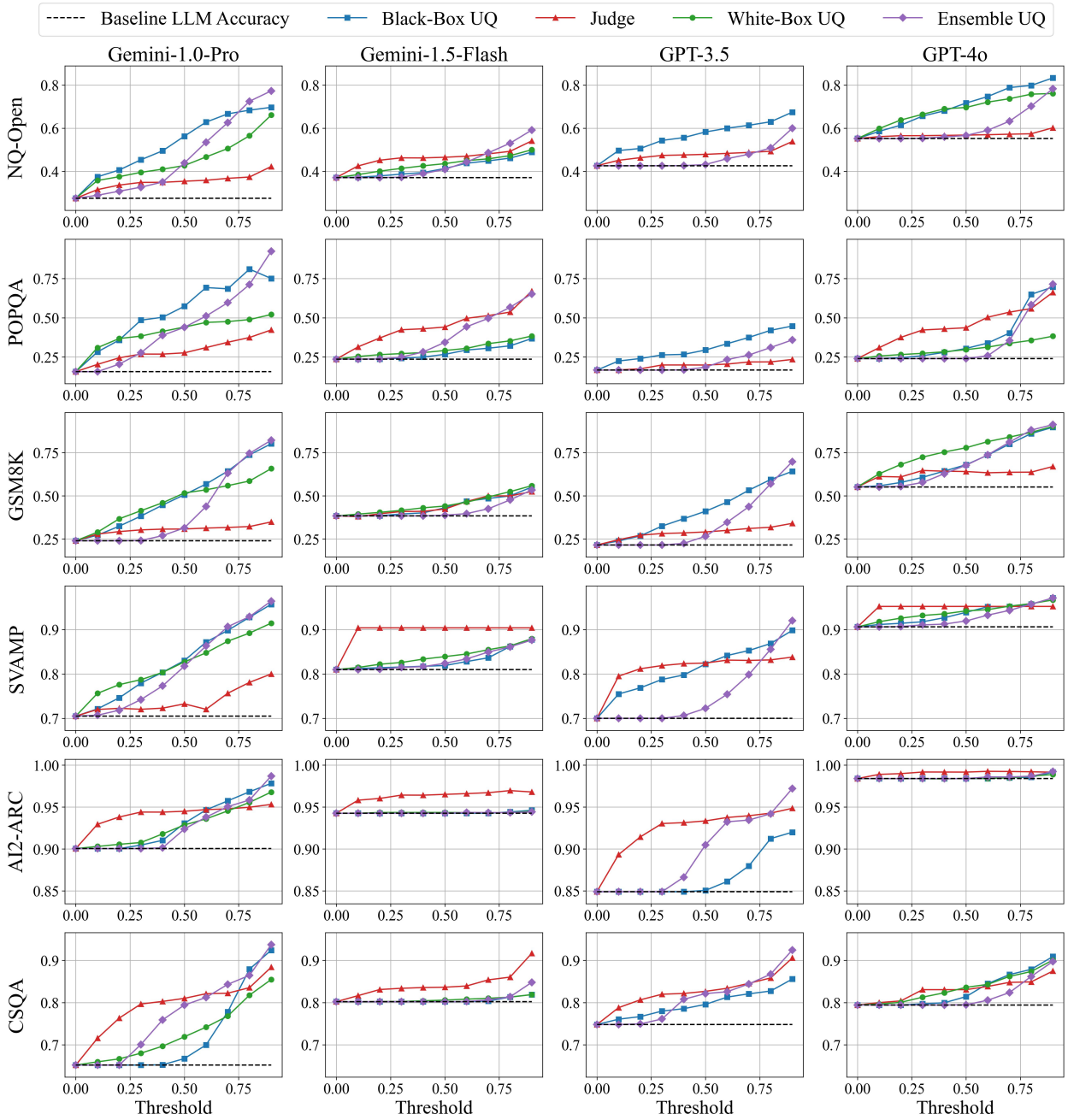

Figure 3: Filtered LLM Accuracy vs. Confidence Threshold (Top per Scorer Type)

Two additional limitations relate to the ensemble approach. First, our experiments consider only linear ensembles. Future work should examine non-linear ensembling and its impact on hallucination detection, e.g., monotonic generalized additive models that retain interpretability while capturing modest non-linear effects, mixture-of-experts, and tree-based ensembles. These may outperform linear averaging when scorer signals interact or exhibit non-linear relationships. Practical trade-offs to study include latency, overfitting risk on small graded sets, and the need for interpretability in high-stakes deployments. Second, tuning the

ensemble requires a graded dataset. For tasks with trivial grading (e.g., arithmetic, multiple-choice, short-answer), where there are well-defined, easy-to-recognize correct answers and comparisons with generated text can be computed automatically, this can be done by sampling a representative set of questions, generating responses, computing the selected UQ scores, and using the automatically derived labels to fit the weights. For tasks with non-trivial grading (e.g., summarization), practitioners can begin with a small human-graded dataset (e.g., a few hundred items); additional labels can then be added incrementally from production logs to further optimize the weights and improve ensemble performance.

6 Conclusions

In this paper, we detail a framework for zero-resource hallucination detection comprised of various black-box UQ, white-box UQ, and LLM-as-a-Judge scorers. To ensure standardized outputs of the scorers, we transform and normalize scorers (if necessary) such that all outputs range from 0 to 1, with higher scores indicating greater confidence in an LLM response. These response-level confidence scores can be used for generation-time hallucination detection across a wide variety of LLM use cases. Additionally, we introduce a novel, ensemble-based approach that leverages an optimized weighted average of any combination of individual confidence scores. Importantly, the extensible nature of our ensemble means that practitioners can include additional scorers as new methods become available. To streamline implementation of the framework, all included scorers can be easily implemented using this paper's companion Python toolkit.

Our experimental evaluation of UQ-based scorers offers clear guidance for practitioners. Ensemble approaches consistently outperform individual methods for hallucination detection, with our findings strongly supporting use-case-specific customization rather than one-size-fits-all solutions. For those without token-probability access, NLI-based approaches typically provide the best black-box performance, though practitioners should be aware that all black-box methods struggle when response variation is limited. Importantly, gains from sampling additional responses diminish as the number of candidate responses increases, offering a practical deployment guideline that balances effectiveness with computational efficiency. Finally, our analysis revealed that a model's accuracy on a specific dataset positively relates to its ability to judge responses to questions from that same dataset, providing a practical criterion for judge selection in evaluation frameworks.

References

Ayush Agrawal, Mirac Suzgun, Lester Mackey, and Adam Tauman Kalai. Do language models know when they're hallucinating references?, 2024. URL https://arxiv.org/abs/2305.18248.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019. doi: 10.1145/3292500.3330701.

Yavuz Faruk Bakman, Duygu Nur Yaldiz, Baturalp Buyukates, Chenyang Tao, Dimitrios Dimitriadis, and Salman Avestimehr. Mars: Meaning-aware response scoring for uncertainty estimation in generative llms, 2024. URL https://arxiv.org/abs/2402.11756.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss (eds.), *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909/.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. Lm vs lm: Detecting factual errors via cross examination, 2023. URL https://arxiv.org/abs/2305.13281.

Jeremy R. Cole, Michael J. Q. Zhang, Daniel Gillick, Julian Martin Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. Selectively answering ambiguous questions, 2023. URL https://arxiv.org/abs/2305.14613.

Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, and Maxim Panov. Fact-checking the output of large language models via token-level uncertainty quantification, 2024. URL https://arxiv.org/abs/2403.04696.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin 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.

Nuno M. Guerreiro, Elena Voita, and André 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.

Hsiu-Yuan Huang, Yutong Yang, Zhaoxi Zhang, Sanwoo Lee, and Yunfang Wu. A survey of uncertainty estimation in llms: Theory meets practice, 2024. URL https://arxiv.org/abs/2410.15326.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023. URL https://arxiv.org/abs/2311.05232.

Mingjian Jiang, Yangjun Ruan, Prasanna Sattigeri, Salim Roukos, and Tatsunori Hashimoto. Graph-based uncertainty metrics for long-form language model outputs, 2024. URL https://arxiv.org/abs/2410.20783.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms, 2024. URL https://arxiv.org/abs/2406.15927.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering, 2019. URL https://arxiv.org/abs/1906.00300.

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

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

Chen Ling, Xujiang Zhao, Xuchao Zhang, Wei Cheng, Yanchi Liu, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Jie Ji, Guangji Bai, Liang Zhao, and Haifeng Chen. Uncertainty quantification for in-context learning of large language models, 2024. URL https://arxiv.org/abs/2402.10189.

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

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

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 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/.

Potsawee Manakul, Adian Liusie, and Mark 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. Introducing gpt-4.5 | openai, 2025. URL https://openai.com/index/introducing-gpt-4-5/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pp. 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

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

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

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

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

Jie Ren, Yao Zhao, Tu Vu, Peter J. Liu, and Balaji Lakshminarayanan. Self-evaluation improves selective generation in large language models, 2023. URL https://arxiv.org/abs/2312.09300.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha 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.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. Commonsenseqa 2.0: Exposing the limits of ai through gamification, 2022. URL https://arxiv.org/abs/2201.05320.

S. M Towhidul Islam Tonmoy, S M Mehedi Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models, 2024. URL https://arxiv.org/abs/2401.01313.

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

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong 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.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models, 2024. URL https://arxiv.org/abs/2404.18796.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models, 2024. URL https://arxiv.org/abs/2306.11698.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms, 2024. URL https://arxiv.org/abs/2306.13063.

Caiqi Zhang, Fangyu Liu, Marco Basaldella, and Nigel Collier. Luq: Long-text uncertainty quantification for llms, 2024. URL https://arxiv.org/abs/2403.20279.

Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, and Luoyi Fu. Enhancing uncertainty-based hallucination detection with stronger focus, 2023. URL https://arxiv.org/abs/2311.13230.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

A LLM-as-a-Judge Prompt

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)
>
> # 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 Ensemble Tuning

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.[17] Given a set of n prompts, denoted as $\mathbf{x}$:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \tag{13}$$

denote corresponding correct reference answers as $\mathbf{y}^*$,

$$\mathbf{y}^* = \begin{pmatrix} y_1^* \\ y_2^* \\ \vdots \\ y_n^* \end{pmatrix}, \tag{14}$$

[17] 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.

original LLM responses as $\mathbf{y}$,

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \tag{15}$$

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

$$\tilde{\mathbf{Y}} = \begin{pmatrix} \tilde{\mathbf{y}}_1 \\ \tilde{\mathbf{y}}_2 \\ \vdots \\ \tilde{\mathbf{y}}_n \end{pmatrix} = \begin{pmatrix} \tilde{y}_{11} & \tilde{y}_{12} & \cdots & \tilde{y}_{1m} \\ \tilde{y}_{21} & \tilde{y}_{22} & \cdots & \tilde{y}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{y}_{n1} & \tilde{y}_{n2} & \cdots & \tilde{y}_{nm} \end{pmatrix}. \tag{16}$$

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

$$\hat{\mathbf{s}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{x}, \mathbf{w}) = \begin{pmatrix} \hat{s}(y_1; \tilde{\mathbf{y}}_1, x_1, \mathbf{w}) \\ \hat{s}(y_2; \tilde{\mathbf{y}}_2, x_2, \mathbf{w}) \\ \vdots \\ \hat{s}(y_n; \tilde{\mathbf{y}}_n, x_n, \mathbf{w}) \end{pmatrix}, \tag{17}$$

$$\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{x}, \mathbf{w}, \tau) = \begin{pmatrix} \hat{h}(y_1; \tilde{\mathbf{y}}_1, x_1, \mathbf{w}, \tau) \\ \hat{h}(y_2; \tilde{\mathbf{y}}_2, x_2, \mathbf{w}, \tau) \\ \vdots \\ \hat{h}(y_n; \tilde{\mathbf{y}}_n, x_n, \mathbf{w}, \tau) \end{pmatrix}, \tag{18}$$

and

$$\mathbf{h}(\mathbf{y}; \mathbf{y}^*, \mathbf{x}) = \begin{pmatrix} h(y_1; y_1^*, x_1) \\ h(y_2; y_2^*, x_2) \\ \vdots \\ h(y_n; y_n^*, x_n) \end{pmatrix}, \tag{19}$$

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:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathcal{W}}{\arg\max}\, \mathcal{S}(\hat{\mathbf{s}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{x}, \mathbf{w}), \mathbf{h}(\mathbf{y}; \mathbf{y}^*, \mathbf{x})), \tag{20}$$

where

$$\mathcal{W} = \{(w_1, ..., w_K) : \sum_{k=1}^{K} w_k = 1, w_k \geq 0 \ \forall \ k = 1, ..., K\} \tag{21}$$

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:

$$\tau^* = \arg\max_{\tau \in (0,1)} \mathcal{B}(\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{x}, \mathbf{w}^*, \tau), \mathbf{h}(\mathbf{y}; \mathbf{y}^*, \mathbf{x})), \tag{22}$$

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:

$$\mathbf{w}^*, \tau^* = \arg\max_{\mathbf{w} \in \mathcal{W}, \tau \in (0,1)} \mathcal{B}(\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{x}, \mathbf{w}, \tau), \mathbf{h}(\mathbf{y}; \mathbf{y}^*, \mathbf{x})), \tag{23}$$

where terms follow the same definitions as above.

C Response Grading

For each task-type, the LLM is instructed to output its response in a specific form. We therefore instantiate $h(\cdot)$ with task-specific graders aligned to that form. Unless noted, grading is case-insensitive and trims whitespace.[18]

Math (numeric answer only). For math questions, the LLM is provided the following instruction: "Return only the numerical answer with no additional text." Let y_i be the model output. Let $\text{int}(y_i)$ extract the leading integer substring from y_i if present; otherwise return $\varnothing$. Let $y_i^* \in \mathbb{Z}$ be the correct integer answer.

$$h_{\text{math}}(y_i; y_i^*) = \begin{cases} 0 & \text{if } \text{int}(y_i) = y_i^*, \\ 1 & \text{otherwise.} \end{cases}$$

Non-numeric or missing integer outputs are graded as incorrect.

Multiple choice (letter only). For multiple choice questions, the LLM is provided the following instruction: "Return only the letter of the response with no additional text or explanation." Let $\text{norm}_{mc}(y_i) \in \{A, B, C, D, E\}$ denote response y_i after normalization to uppercase and trimming. Let $y_i^* \in \{A, B, C, D, E\}$ be the correct letter.

$$h_{\text{mc}}(y_i; y_i^*) = \begin{cases} 0 & \text{if } \text{norm}_{mc}(y_i) = y_i^*, \\ 1 & \text{otherwise.} \end{cases}$$

Any invalid response based on the provided instruction, i.e. $\text{norm}_{mc}(y_i) \notin \{A, B, C, D, E\}$, is graded as incorrect.

Short-answer (keyword match). Lastly, for short-answer questions, the LLM is instructed as follows: "Return only the answer as concisely as possible without providing an explanation." Let $\mathbf{y}_i^* = \{y_{i1}^*, \ldots, y_{iA}^*\}$ be the set of acceptable answers.[19] Define a normalization function $\text{norm}_{sa}(y_i)$ that lowercases and trims y_i. Let $\text{contains}(u, v)$ indicate that string u contains string v as a contiguous substring.

$$h_{\text{sa}}(y_i; \mathbf{y}_i^*) = \begin{cases} 0 & \text{if } \exists\, y_i^* \in \mathbf{y}_i^* \text{ s.t. } \text{contains}(\text{norm}_{sa}(y_i), \text{norm}_{sa}(y_i^*)), \\ 1 & \text{otherwise.} \end{cases}$$

If output includes additional text, grading is based on the presence of any acceptable answer string.

[18] These automatic graders may not be perfectly accurate in every instance; residual errors can arise when model outputs deviate from the specified response format. Their effectiveness therefore depends on instruction adherence, and they should be viewed as practical proxies for answer-key–based factual correctness.

[19] The short-answer datasets contain multiple versions of the same correct answer to allow for phrasing flexibility. For example, acceptable answers to the question "What is Tobias Lindholm's occupation?" include {"film director", "movie director", "director", "motion picture director", "screenwriter", "scenarist", "writer", "screen writer", "script writer", "scriptwriter"}.

D Additional Experiments: Number of Candidate Responses vs. Black-Box UQ Performance

To investigate the effect of number of candidate responses m on the performance of black-box scorers, we re-compute all black-box confidence scores for $m = 1, 3, 5, 10, 15$ for each of our 24 LLM-dataset scenarios. We compute the scorer-specific AUROC value for each value of m and evaluate the impact of number of candidate responses on hallucination detection performance. These results are depicted in Figure 4.

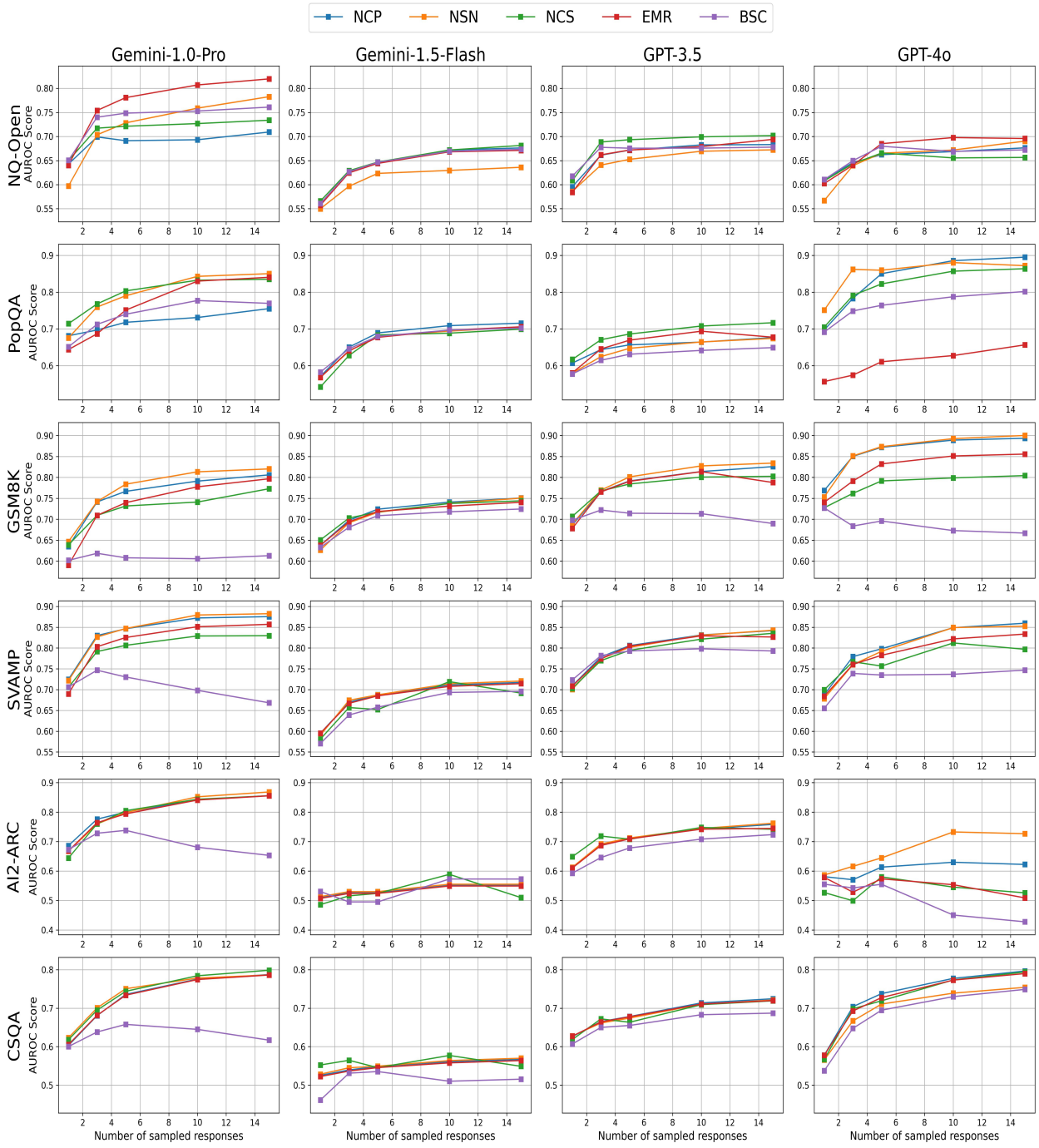

Figure 4: Hallucination Detection AUROC by Number of Sampled Responses

Overall, the results indicate that hallucination detection performance of the black-box scorers improves considerably with number of candidate responses m. For instance, hallucination detection AUROC of the various black-box scorers on GPT-4o responses on CSQA improve from 0.54-0.57 with $m = 1$ to 0.75-0.8 with $m = 15$. In the vast majority of our experimental scenarios, these performance improvements occur approximately monotonically with diminishing returns to higher m, consistent with findings from previous studies (Kuhn et al., 2023; Manakul et al., 2023; Lin et al., 2024; Farquhar et al., 2024). With the exception of BSC, this general trend is consistent in 21 of the 24 scenarios.

We observe two notable exceptions to this trend. First, BSC does not exhibit improved performance for higher m in seven of the scenarios. This finding of overall less improvement in hallucination detection with higher m for BSC is consistent with the experiments of Manakul et al. (2023) and is likely due to the token-wise nature of the comparisons.

Table 3: 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

Second, for Gemini-1.5-Flash responses on the AI2-ARC and CSQA datasets, we observe both poor black-box scorer performance (0.46-0.56) and a lack of improvement when increasing m. To investigate this finding, we computed the average exact match rate (EMR) across all 24 scenarios, with results presented in Table 3. Notably, Gemini-1.5-Flash exhibits remarkably high average EMR values on AI2-ARC (1.00) and CSQA (0.99), indicating almost no variation in the generated responses. This lack of diversity explains why increasing m yields negligible improvement for these scenarios: additional samples provide virtually no new information.

More broadly, Gemini-1.5-Flash demonstrates consistently higher EMR across all benchmarks compared to other models. Accordingly, black-box hallucination detection is no better or worse on these Gemini-1.5-Flash responses compared to other LLM responses for the same dataset. This finding suggests that the effectiveness of black-box hallucination detection is fundamentally limited by response diversity and is consistent with findings by Kuhn et al. (2023), who find performance of various black-box scorers is considerably worse on smaller models that exhibit less response variation.