Uncertainty Quantification for Language Models: Black-Box, White-Box, Judges, and Ensemble Methods

Dylan Bouchard[1] * **Mohit Singh Chauhan**[1]
[1]CVS Health®

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 propose 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 introduce an 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 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 generally surpasses its individual components and outperforms existing 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 an increasing need to monitor these systems for the accuracy and factual correctness of model outputs. Mistakes in these sensitive use cases can lead to high financial costs and reputation 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). Recent models, such as OpenAI's GPT-4.5, still hallucinate as often as 37.1% [Barrabi, 2025].

One approach to hallucination detection is human-in-the-loop, where a human reviewer fact-checks LLM outputs. However, LLM systems are often implemented at a scale that is prohibitively large for exhaustive human review. In high-risk LLM applications, sampling-based human-in-the-loop, where a human manually reviews a random subset of LLM outputs, is unlikely to suffice. This motivates the need for an automated, response-level method to identify LLM outputs that are most likely to contain hallucinations. Such a technique enables filtering or targeted human review of responses with low confidence.

Computational 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

*Correspondence to dylan.bouchard@cvshealth.com

of systems already deployed to production. In contrast, techniques that compare source content to generated content or quantify uncertainty can compute response-level scores 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 propose 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 black-box UQ, white-box UQ, and LLM-as-a-Judge methods, transforming them as necessary into standardized confidence scores on a 0-to-1 scale. For improved customization, we propose a tunable ensemble approach that incorporates any combination of the individual confidence scores. 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 conduct a comprehensive evaluation of the scorers on a range of LLM question-answering benchmarks, yielding several key insights. Notably, black-box and white-box UQ scorers consistently outperform LLM-as-a-Judge. However, the top-performing individual scorers vary significantly across datasets, highlighting the importance of tailoring approaches to specific use cases. Furthermore, our tunable ensemble generally surpasses its individual components, demonstrating the benefits of customized hallucination detection strategies.

2 Related Work

In this section, we discuss a collection of studies that propose zero-resource hallucination detection techniques. These techniques compute response-level confidence or uncertainty scores without requiring access to source content, external resources (such as the internet), or ground truth texts. We consider four types of zero-resource hallucination detection: black-box UQ, white-box UQ, LLM-as-a-Judge, and ensemble methods.

2.1 Black-Box UQ

Several studies have proposed methods for quantifying uncertainty using semantic similarity between an original LLM response and a set of candidate responses generated from the same prompt. These methods typically involve pairwise comparison using exact match comparisons, text similarity metrics, model-based text similarity, or natural language inference (NLI) models.

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. While these metrics have the advantage of being intuitive, they have two notable disadvantages. First, minor phrasing differences are penalized, even if two responses have the same meaning. Second, these metrics are poorly suited for tasks that do not have a unique correct answer, such as summarization tasks.

Text similarity metrics assess response consistency in 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. BLEURTScore [Sellam et al., 2020] and BARTScore [Yuan et al., 2021] are notable alternatives.

Lastly, 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 and a 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. 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.

2.2 White-Box UQ

In contrast to black-box UQ methods, white-box UQ methods require access to underlying token probabilities associated with the LLM's generated responses. Since these methods typically involve performing simple arithmetic on a generation's token probabilities, they have the advantage of not impacting latency. Additionally, because these token-based scores can be computed from a single response, they do not add any cost beyond regular cost of generation. Despite these advantages, these approaches have limited compatibility, as many APIs for querying LLMs do not provide access to token probabilities.

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 responses, which has the advantage of being bounded and easy to interpret.

2.3 LLM-as-a-Judge

Another popular approach for uncertainty quantification is LLM-as-a-judge [Gu et al., 2025], where either the same LLM that was used for generating the original responses or a different LLM is asked to form a judgment about a pre-generated response. The nature of the judgment may be a numerical score [Xiong et al., 2024a, Kadavath et al., 2022, Jones et al., 2024, Li et al., 2023, Zhu et al., 2023, Xiong et al., 2024b, Bai et al., 2023] or a binary yes/no [Shinn et al., 2023, Tian et al., 2024, Sun et al., 2024].

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. [2024a] 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.

2.4 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 observed consistency and self-reflection certainty.

Fallah et al. [2024] leverage an ensemble of judges to evaluate uncertainty of LLM responses. Specifically, they propose three ensemble variations, each comprised of three LLM judges. The first two involve having two LLMs comment on a response's correctness and having one of those two commentator LLMs form a judgment based on both sets of comments. The third involves having two commentator LLMs with a third separate LLM forming the final judgment based on the comments of the first two. Similarly, 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 $\hat{s} : \mathcal{Y} \to [0, 1]$.

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. Below, we provide details of these various scorers.[2]

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 involves generating m 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:

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

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

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

[2023], also conducts pairwise comparison between the original response and each candidate response. In particular, a natural language inference (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:

$$NCP(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{m} \sum_{j=1}^{m} (1 - p_j) \tag{4}$$

$$p_j = \frac{\eta(y_i, \tilde{y}_{ij}) + \eta(\tilde{y}_{ij}, y_i)}{2}. \tag{5}$$

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.

Normalized Semantic Negentropy. Semantic entropy (SE), proposed by Farquhar et al. [2024], exploits variation in multiple responses to compute a measure of response volatility. In contrast to the EMR and NCP, semantic entropy does not distinguish between an original response and candidate responses. Instead, this approach computes a single metric value on a list of responses generated from the same prompt. Under this approach, responses are clustered using an NLI model based on mutual entailment. We consider the discrete version of SE, where the final set of clusters is defined as follows:

$$SE(y_i; \tilde{\mathbf{y}}_i) = -\sum_{C \in \mathcal{C}} P(C|y_i, \tilde{\mathbf{y}}_i) \log P(C|y_i, \tilde{\mathbf{y}}_i), \tag{6}$$

where $P(C|y_i, \tilde{\mathbf{y}}_i)$ denotes the probability a randomly selected response $y \in \{y_i\} \cup \tilde{\mathbf{y}}_i$ belongs to cluster C, and $\mathcal{C}$ denotes the full set of clusters of $\{y_i\} \cup \tilde{\mathbf{y}}_i$.[3]

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):

$$NSN(y_i; \tilde{\mathbf{y}}_i) = 1 - \frac{SE(y_i; \tilde{\mathbf{y}}_i)}{\log m}, \tag{7}$$

where $\log m$ is included to normalize the support.

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, recall, and F1-scores between two tokenized texts $\mathbf{t}, \mathbf{t}'$ are respectively defined as follows:

$$BertPrec(\mathbf{t}, \mathbf{t}') = \frac{1}{|\mathbf{t}|} \sum_{t \in \mathbf{t}} \max_{t' \in \mathbf{t}'} \mathbf{e} \cdot \mathbf{e}' \tag{8}$$

$$BertRec(\mathbf{t}, \mathbf{t}') = \frac{1}{|\mathbf{t}'|} \sum_{t' \in \mathbf{t}'} \max_{t \in \mathbf{t}} \mathbf{e} \cdot \mathbf{e}' \tag{9}$$

$$BertFScore(\mathbf{t}, \mathbf{t}') = 2 \frac{BertPrec(\mathbf{t}, \mathbf{t}') BertRec(\mathbf{t}, \mathbf{t}')}{BertPrec(\mathbf{t}, \mathbf{t}') + BertRec(\mathbf{t}, \mathbf{t}')}, \tag{10}$$

where e, e' respectively correspond to t, t'. We compute our BERTScore-based confidence scores as follows:

$$BertConfidence(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{m} \sum_{j=1}^{m} BertFScore(y_i, \tilde{y}_{ij}), \tag{11}$$

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

[3] 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.

BLEURT. Similar to BERTScore, BLEURT is a BERT-based scorer based for evaluating text similarity [Sellam et al., 2020].[4] In contrast to the aforementioned scorers, BLEURT is specifically pre-trained and fine-tuned to learn human judgments of text similarity.[5] Our BLEURT confidence score is the average BLEURT value across pairings of the original response with all candidate responses:

$$BLEURTConfidence(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{m} \sum_{j=1}^{m} BLEURT(y_i, \tilde{y}_{ij}). \tag{12}$$

Normalized Cosine Similarity. Lastly, we include normalized cosine similarity (NCS) to measure semantic similarity [Shorinwa et al., 2024]. This scorer 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. The average cosine similarity across pairings of the original response with all candidate responses is given as follows:

$$CS(y_i; \tilde{\mathbf{y}}_i) = \frac{1}{m} \sum_{i=1}^{m} \frac{\mathbf{V}(y_i) \cdot \mathbf{V}(\tilde{y}_{ij})}{\|\mathbf{V}(y_i)\| \|\mathbf{V}(\tilde{y}_{ij})\|}. \tag{13}$$

To ensure a standardized support of $[0, 1]$, we normalize cosine similarity to obtain confidence scores as follows:

$$NCS(y_i; \tilde{\mathbf{y}}_i) = \frac{CS(y_i; \tilde{\mathbf{y}}_i) + 1}{2}. \tag{14}$$

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.

Length-Normalized Token Probability. Let the tokenization 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:

$$LNTP(y_i) = \prod_{t \in y_i} p_t^{L_i}, \tag{15}$$

where p_t denotes the token probability for token t. 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:

$$MTP(y_i) = \min_{t \in y_i} p_t, \tag{16}$$

where t and p_t follow the same definitions as above.

3.4 LLM-as-a-Judge Scorers

Below we outline two LLM-as-a-Judge scorers. We follow the approach proposed by Chen and Mueller [2023] in which an LLM is instructed to score a question-answer concatenation as either *incorrect*, *uncertain*, or *correct* using a carefully constructed prompt. These categories are respectively mapped to numerical scores of 0, 0.5, and 1. We denote the LLM-as-a-judge scorers as $J : \mathcal{Y} \to \{0, 0.5, 1\}$. Formally, we can write these scorer functions as follows:

$$J(y_i) = \begin{cases} 0 & \text{LLM states response is incorrect} \\ 0.5 & \text{LLM states that it is uncertain} \\ 1 & \text{LLM states response is correct.} \end{cases} \tag{17}$$

[4] BLEURT is an acronym for Bilingual evaluation Understudy with Representations from Transformers.

[5] We use the recommended BLEURT checkpoint of `BLEURT-20`.[Sellam et al., 2020]

In a slight modification from the prompt used by Chen and Mueller [2023], we use the following prompt:

> Question: [question], Proposed Answer: [answer].
>
> Your task is to look at the question and answer provided and determine if the answer is correct. You are to respond with ONLY one of: "Correct", "Incorrect", or "I am not sure". YOUR ANSWER MUST ONLY CONTAIN ONE OF "Correct", "Incorrect", or "I am not sure". DO NOT ANSWER THE QUESTION AGAIN. ONLY DETERMINE IF THE ANSWER TO THE QUESTION IS "Correct", "Incorrect", or "I am not sure".

The capitalization and repeated instructions, inspired by Wang et al. [2024], are included to ensure the LLM correctly follows instructions. The instruction part of the prompt is also used as the LLM's system prompt for the generation of LLM judgments.

Self-Judge. Under the self-judge approach, the same LLM used to generate the original responses is used to score the responses with the aforementioned approach. Importantly, the query asking the LLM to judge the response is passed independently and does not include the chat history including the original prompt and response.

External Judge. A generalization of the self-reflection certainty approach proposed by Chen and Mueller [2023], our external LLM-as-a-judge approach follows the same setup as the self-judge, but instead leverages a *different* LLM to classify the response as correct, incorrect, or uncertain. Note that multiple LLM-as-a-judge scorers can be used if multiple LLMs are available to the practitioner.

3.5 Ensemble Scorer

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

$$\hat{s}(y_i; \tilde{\mathbf{y}}_i, \mathbf{w}) = \sum_{k=1}^{K} w_k \hat{s}_k(y_i; \tilde{\mathbf{y}}_i), \tag{18}$$

where $\mathbf{w} = (w_1, ..., w_K)$, $\sum_{k=1}^{K} w_k = 1$, and $w_k \in [0, 1]$ for $k = 1, ..., K$. Note that although we write each classifier to be parameterized by the set of candidate responses, some of the classifiers depend only on the original response.

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.[6] For a set of n prompts, we denote the vector of original responses as $\mathbf{y}$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \tag{19}$$

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

$$\tilde{\mathbf{Y}} = \begin{pmatrix} \tilde{\mathbf{y}}_1 \\ \tilde{\mathbf{y}}_2 \\ \vdots \\ \tilde{\mathbf{y}}_n \end{pmatrix} = \begin{pmatrix} \tilde{y}_{11} & \tilde{y}_{12} & \cdots & \tilde{y}_{1m} \\ \tilde{y}_{21} & \tilde{y}_{22} & \cdots & \tilde{y}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{y}_{n1} & \tilde{y}_{n2} & \cdots & \tilde{y}_{nm} \end{pmatrix}. \tag{20}$$

[6] 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.

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

$$\hat{\mathbf{s}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}) = \begin{pmatrix} \hat{s}(y_1; \tilde{\mathbf{y}}_1, \mathbf{w}) \\ \hat{s}(y_2; \tilde{\mathbf{y}}_2, \mathbf{w}) \\ \vdots \\ \hat{s}(y_n; \tilde{\mathbf{y}}_n, \mathbf{w}) \end{pmatrix}, \tag{21}$$

$$\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}, \tau) = \begin{pmatrix} \hat{h}(y_1; \tilde{\mathbf{y}}_1, \mathbf{w}, \tau) \\ \hat{h}(y_2; \tilde{\mathbf{y}}_2, \mathbf{w}, \tau) \\ \vdots \\ \hat{h}(y_n; \tilde{\mathbf{y}}_n, \mathbf{w}, \tau) \end{pmatrix}, \tag{22}$$

and

$$\mathbf{h}(\mathbf{y}) = \begin{pmatrix} h(y_1) \\ h(y_2) \\ \vdots \\ h(y_n) \end{pmatrix}. \tag{23}$$

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:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathcal{W}}{\arg\max}\, \mathcal{S}(\hat{\mathbf{s}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}), \mathbf{h}(\mathbf{y})), \tag{24}$$

where

$$\mathcal{W} = \{(w_1, ..., w_K) : \sum_{k=1}^{K} w_k = 1, w_k \geq 0 \; \forall \; k = 1, ..., K\} \tag{25}$$

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:

$$\tau^* = \underset{\tau \in (0,1)}{\arg\max}\, \mathcal{B}(\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}^*, \tau), \mathbf{h}(\mathbf{y})), \tag{26}$$

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:

$$\mathbf{w}^*, \tau^* = \underset{\mathbf{w} \in \mathcal{W}, \tau \in (0,1)}{\arg\max}\, \mathcal{B}(\hat{\mathbf{h}}(\mathbf{y}; \tilde{\mathbf{Y}}, \mathbf{w}, \tau), \mathbf{h}(\mathbf{y})), \tag{27}$$

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

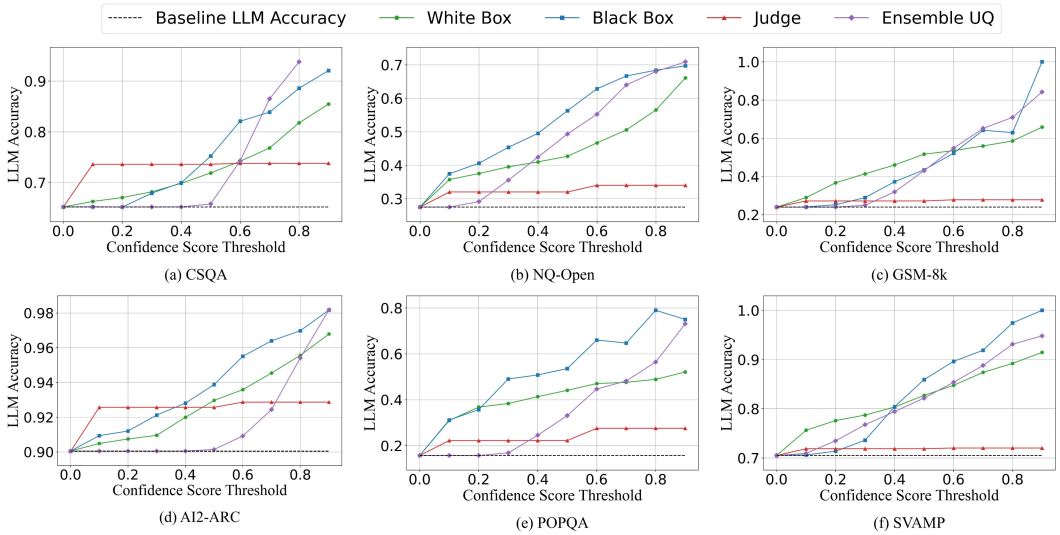


Figure 1: Filtered Accuracy@τ Results: Performance of Ensemble vs. BlackBox vs. WhiteBox vs. Judges for Gemini Responses

4 Experiments

4.1 Experiment Setup.

We conduct a series of experiments to assess the hallucination detection performance of our 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 *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 first sample 1000 question for each of our six benchmarks. For each question, we generate an original response and $m = 15$ candidate responses using both `gpt-3.5-16k-turbo` and `gemini-1.0-pro`. For each response, we use the corresponding candidate responses to compute the full suite of black-box UQ scores. We also compute self-judge and external judge scores for each response generated by both models. Lastly, we compute white-box scores for the `gemini-1.0-pro` responses.[7] We evaluate the hallucination detection performance of all individual scorers as well our ensemble scorer for each of the benchmarks using various metrics.

4.2 Experiment Results.

Filtered Accuracy@τ To assess the reliability of the confidence scores, we first compute model accuracy on the subset of model responses having confidence scores exceeding a specified threshold τ. Since the model accuracy depends on the choice of the threshold τ, we repeat the calculation for multiple values of the confidence score threshold ($\tau \in \{0, 0.1, \ldots, 0.9\}$). Note that accuracy at $\tau = 0$ uses the full sample without score-based filtering.

The full set of filtered accuracy@τ results are displayed in Figure 1.[8] For each benchmark, we plot values for the highest performing white-box, black-box, and LLM-as-a-Judge scorers, along with the ensemble scorer. Across all benchmarks, we see LLM accuracy increases with the threshold approximately monotonically for white-box, black-box and ensemble scorers. We see that white-box, black-box, and ensemble scorers consistently outperform LLM-as-a-Judge across all benchmarks. Overall, these results demonstrate the ability of white-box and black-box scorers to improve LLM accuracy using confidence score-based filtering.

[7] Our instance of `gpt-3.5-16k-turbo` did not support token probability access. Hence, we were unable to compute white-box UQ scores for the `gpt-3.5-16k-turbo` responses.

[8] The figures depicting results for the `gpt-3.5-16k-turbo` experiments are contained in Appendix A.

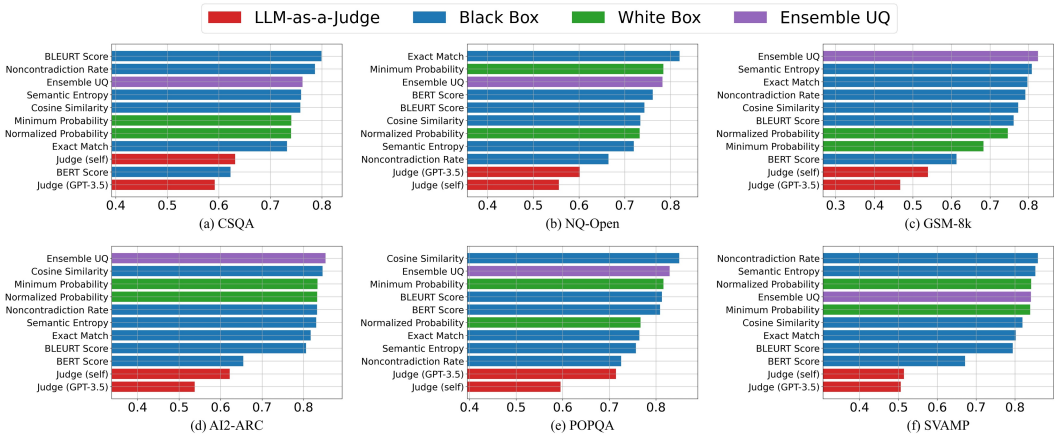


Figure 2: Gemini ROC-AUC score: Ensemble vs. BlackBox vs. WhiteBox vs. Judges

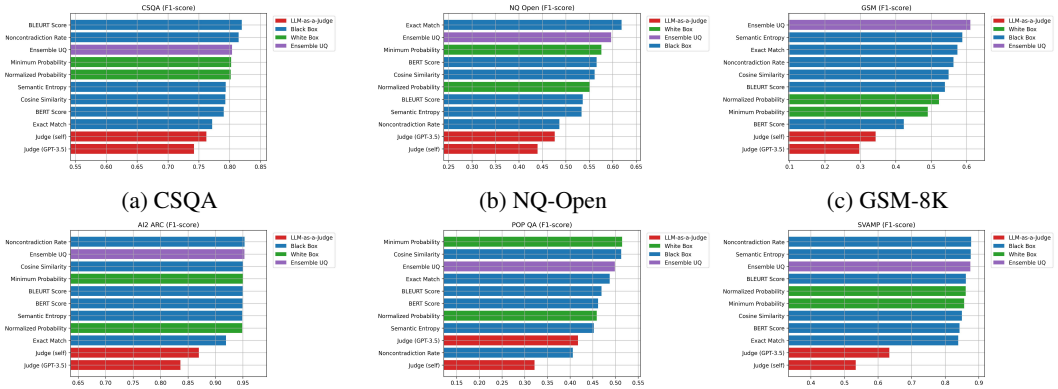


Figure 3: Gemini F1-Score: Ensemble vs. BlackBox vs. WhiteBox vs. Judges

Threshold-Agnostic Evaluation To further assess the performance of our scorers as hallucination classifiers, we evaluate the performance of the confidence scores in a threshold-agnostic fashion. Under this setting, we use the AUROC-optimized ensemble weights and compute the score-specific AUROC using 5-fold cross validation.

These results are depicted in Figure 2. We find that black-box and white-box scorers outperform LLM-as-a-Judge, with the two LLM-as-a-Judge scorers ranking last for five of six benchmarks. Among white-box and black-box scorers, we see substantial variation in which scorers perform best across benchmarks. Last, we find that our ensemble approach typically performs as well or better than its individual components. Overall, these findings demonstrate the use-case-specific nature of hallucination detection performance and the benefits of customizing one's hallucination detection strategy.

Threshold-Optimized Evaluation Lastly, we evaluate the various scorers using threshold-dependent metrics. 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 Section 3.5. As with AUROC, we compute the score-specific F1-scores using 5-fold cross validation.

The scorer-specific F1-scores are depicted in Figure 3. Again, we find that black-box and white-box scorers substantially outperform LLM-as-a-Judge, with the two LLM-as-a-Judge scorers ranking in the bottom three for all benchmarks. As with AUROC, we find that the top-performing scorers vary by dataset, further indicating the use-case-specific nature of hallucination detection performance. Again, our ensemble scorer typically performs as well or better than its individual components.

5 Discussion

Choosing Among Scorers Choosing the right confidence score for an LLM system depends on several factors, including API support, latency requirements, and the availability of graded datasets. First, practitioners should consider whether the API supports access to token probabilities in LLM generations. If so, white-box scorers can be implemented without adding latency or generation costs. However, if the API does not provide 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 using higher-latency black-box scorers such as semantic entropy, noncontradiction probability, BERTScore, and BLEURT Score. Instead, they can opt for faster black-box scorers or use LLM-as-a-Judge. On the other hand, if latency is not a concern, any of the black-box scorers can be used.

Finally, if a graded dataset is available, practitioners can ensemble various confidence scores to improve hallucination detection performance. As outlined in Section 3.5, a graded dataset can be used to tune an ensemble classifier, which can provide more accurate confidence scores than individual scorers. By considering these factors and choosing the right confidence score, practitioners can optimize 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. Our experimental evaluations of filtered accuracy@τ demonstrate the efficacy of this approach, illustrating the strong correlation between confidence scores and LLM accuracy. Second, practitioners may also consider implementing 'targeted' human-in-the-loop practices. Under this approach, manual review would happen for any response that yields a low confidence score. Again, our filtered accuracy@τ evaluations indicate that this approach will offer far more efficient review compared to arbitrary human-in-the-loop, where random samples of responses are manually reviewed. Additionally, this approach helps address scenarios where exhaustive human-in-the-loop is infeasible due to the scale of responses being generated. Lastly, confidence scores can be used for pre-deployment diagnostics, providing practitioners insights into the types of questions on which their LLM is performing worst.

6 Conclusions

In this paper, we construct a framework for zero-resource hallucination detection comprised of various black-box, white-box, and LLM-as-a-Judge UQ-based scorers. To ensure standardized outputs of the scorers, we transform and normalize scorers 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 effective 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.

We conduct an extensive set of experiments using various question-answering benchmarks to evaluate the effectiveness of the various scorers. We find that, in general, white-box and black-box scorers outperform LLM-as-a-Judge. Among white-box and black-box scorers, we find that the highest performing scorer tends to depend on the dataset and LLM being used. Lastly, we find that ensembles typically outperform their individual components in detecting hallucinations. These findings underscore the benefits of customized hallucination detection methods for use-case-specific optimization.

Acknowledgements

We wish to thank Xue (Crystal) Gu, Piero Ferrante, Blake Aber, David Skarbrevik, Matthew Churgin, Saicharan Sirangi, Erik Widman, and Zeya Ahmad for their helpful suggestions.

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.

Y. Bai, J. Ying, Y. Cao, X. Lv, Y. He, X. Wang, J. Yu, K. Zeng, Y. Xiao, H. Lyu, J. Zhang, J. Li, and L. Hou. Benchmarking foundation models with language-model-as-an-examiner, 2023. URL https://arxiv.org/abs/2306.04181.

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/.

T. Barrabi. Sam altman's openai launches gpt-4.5 with fewer "hallucinations" as ai race heats up, Feb 2025. URL https://www.yahoo.com/tech/sam-altman-openai-launches-gpt-173141886.html?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAKonZVGZuwWeT6ZwH3XIsVwj7RAB11l63dRQUOKSUwI9YuoNXAlUUY_24fy2bXL_JZ1R2UxTItFaGxfJ_MkvEW9Fu--2Isjs4_pl4rqZuU2MVYzbX97XzmZxIx0ItaglG9P57dMYHkjKTtfNqT-BHTRbbefpfu

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.

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.

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.

P. Fallah, S. Gooran, M. Jafarinasab, P. Sadeghi, R. Farnia, A. Tarabkhah, Z. S. Taghavi, and H. Sameti. SLPL SHROOM at SemEval2024 task 06 : A comprehensive study on models ability to detect hallucination. In A. K. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, and A. Rosá, editors, *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1148–1154, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.semeval-1.167. URL https://aclanthology.org/2024.semeval-1.167/.

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.

J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Y. Wang, W. Gao, L. Ni, and J. Guo. A survey on llm-as-a-judge, 2025. URL https://arxiv.org/abs/2411.15594.

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. Jones, L. Mo, E. Fosler-Lussier, and H. Sun. A multi-aspect framework for counter narrative evaluation using large language models, 2024. URL https://arxiv.org/abs/2402.11676.

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.

J. Li, S. Sun, W. Yuan, R.-Z. Fan, H. Zhao, and P. Liu. Generative judge for evaluating alignment, 2023. URL https://arxiv.org/abs/2310.05470.

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.

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.

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.

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

T. Sellam, D. Das, and A. P. Parikh. Bleurt: Learning robust metrics for text generation, 2020. URL https://arxiv.org/abs/2004.04696.

N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

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.

J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H.-Y. Shum, and J. Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph, 2024. URL https://arxiv.org/abs/2307.07697.

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.

Y. Tian, A. Ravichander, L. Qin, R. L. Bras, R. Marjieh, N. Peng, Y. Choi, T. L. Griffiths, and F. Brahman. Macgyver: Are large language models creative problem solvers?, 2024. URL https://arxiv.org/abs/2311.09682.

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.

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.

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, 2024a. URL https://arxiv.org/abs/2306.13063.

T. Xiong, X. Wang, D. Guo, Q. Ye, H. Fan, Q. Gu, H. Huang, and C. Li. Llava-critic: Learning to evaluate multimodal models, 2024b. URL https://arxiv.org/abs/2410.02712.

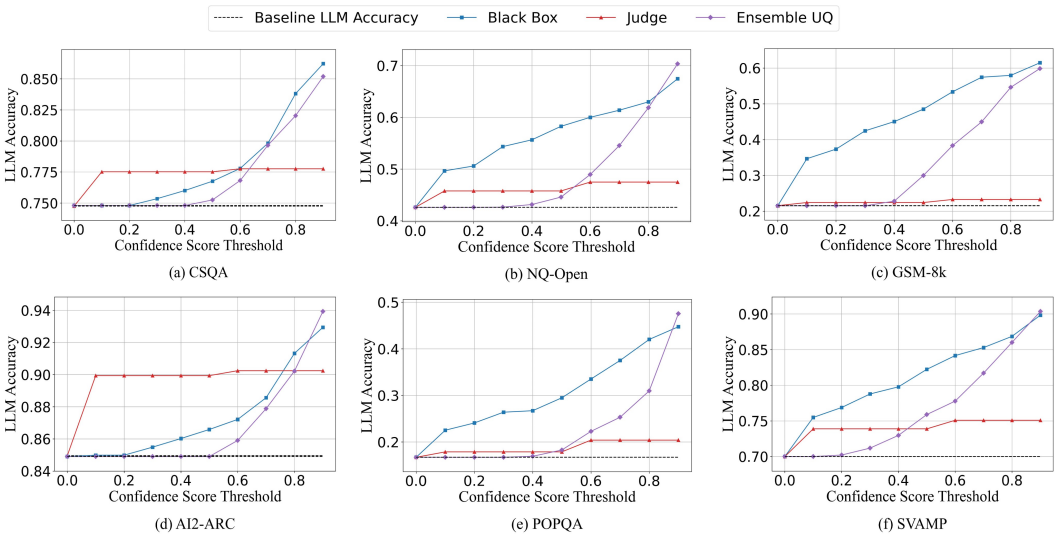
W. Yuan, G. Neubig, and P. Liu. Bartscore: Evaluating generated text as text generation, 2021. URL https://arxiv.org/abs/2106.11520.

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.

L. Zhu, X. Wang, and X. Wang. Judgelm: Fine-tuned large language models are scalable judges, 2023. URL https://arxiv.org/abs/2310.17631.

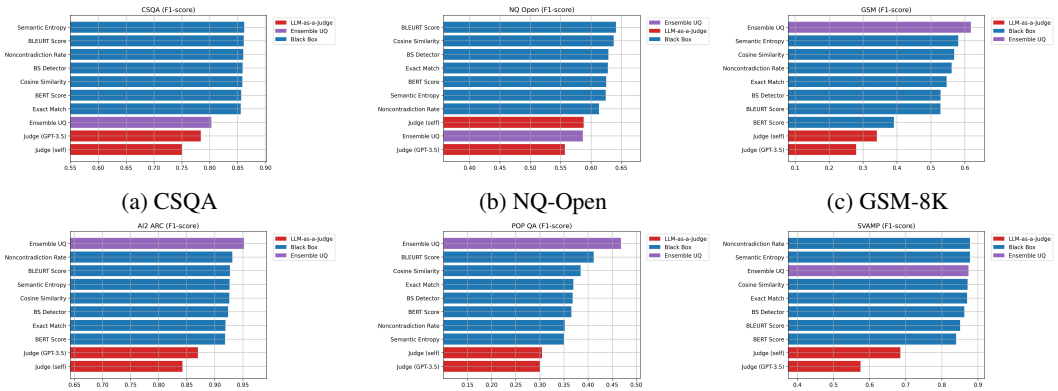
A Supplemental Figures



(a) CSQA (b) NQ-Open (c) GSM-8k

(d) AI2-ARC (e) POPQA (f) SVAMP

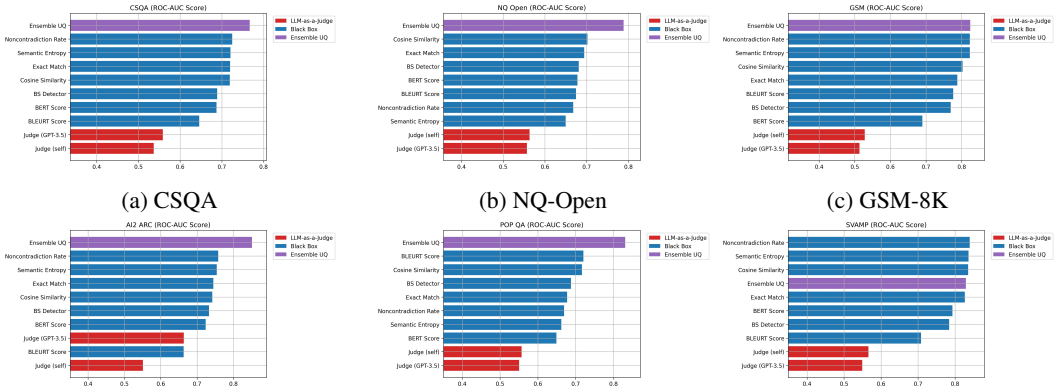
Figure 4: GPT Filtered Accuracy: Ensemble vs. BlackBox vs. Judges



(a) CSQA (b) NQ-Open (c) GSM-8K

(d) AI2-ARC (e) POPQA (f) SVAMP

Figure 5: GPT3.5 F1-Score: Ensemble vs. BlackBox vs. Judges



(a) CSQA (b) NQ-Open (c) GSM-8K

(d) AI2-ARC (e) POPQA (f) SVAMP

Figure 6: GPT ROC-AUC score: Ensemble vs. BlackBox vs. Judges