LLM Uncertainty Cheatsheet

Yukai Song

July 3, 2025

What is Black-Box UQ?

A method to estimate an LLM's **confidence** in its response **without accessing internal probabilities**. It works by generating **multiple candidate responses** to the same prompt and evaluating **semantic consistency** across them.

Core Intuition

"If the model says different things every time you ask it the same question, it probably doesn't know the answer."

Scorer Types, Formulas, and Examples

1. Exact Match Rate (EMR)

Formula:

$$\text{EMR}(y_i) = \frac{1}{m} \sum_{j=1}^{m} I(y_i = \tilde{y}_{ij})$$

Notation:

- y_i: Ground-truth reference sentence for instance i
- $\tilde{y}_{ij}$: j-th candidate response generated by the model
- m: Total number of candidate responses
- $I(\cdot)$: Indicator function that returns 1 if the condition is true, 0 otherwise

Example:

- **Reference:** `"Paris is the capital of France"`
- **Candidates:**
  1. `"Paris is the capital of France"` (Exact Match)
  2. `"Paris is the capital of France"` (Exact Match)
  3. `"Paris is the capital of France"` (Exact Match)
  4. `"The capital of France is Paris"` (Paraphrase)
  5. `"France's capital city is Paris"` (Paraphrase)
- **EMR:** $3/5 = 0.6$

Limitation:
Over-penalizes harmless paraphrasing. Fails to credit semantically equivalent but syntactically different responses.

2. Non-Contradiction Probability (NCP)

Formula:

$$\text{NCP}(y_i) = \frac{1}{m} \sum_{j=1}^{m} \left(1 - \frac{\eta(y_i, \tilde{y}_{ij}) + \eta(\tilde{y}_{ij}, y_i)}{2}\right)$$

Notation:

- y_i: Ground-truth reference sentence for instance i
- $\tilde{y}_{ij}$: j-th candidate response
- m: Number of candidate responses
- $\eta(a, b)$: Probability that sentence a contradicts sentence b, as predicted by a Natural Language Inference (NLI) model

Explanation:
We use an NLI model to compute how likely a candidate contradicts the reference and vice versa. A high NCP value indicates that most candidates are consistent with the reference.

Example:

- **Reference:** `"Obama was born in Hawaii"`
- **Candidates:**
  1. `"Obama was born in Kenya"`
     - $\eta(\text{ref}, \text{cand}) \approx 0.9$
     - $\eta(\text{cand}, \text{ref}) \approx 0.9$
     - $1 - \frac{0.9+0.9}{2} = 0.1$
  2. `"Obama was born in Hawaii in 1961"`
     - $\eta(\text{ref}, \text{cand}) \approx 0.1$
     - $\eta(\text{cand}, \text{ref}) \approx 0.1$
     - $1 - \frac{0.1+0.1}{2} = 0.9$
- **NCP:** $\frac{0.1+0.9}{2} = 0.5$

Limitation:
Depends heavily on the accuracy of the underlying NLI model; subtle contradictions or paraphrased facts may be missed or misclassified.

3. Normalized Semantic Negentropy (NSN)

Formula:

$$\text{NSN}(y_i) = 1 - \frac{H(\text{clusters})}{\log(m+1)}$$

Notation:

- m: Number of model-generated candidate responses.
- Clusters: Groups of semantically similar candidates, obtained via clustering (e.g., NLI-based or embedding similarity).
- $H = -\sum_k P_k \log P_k$: Shannon entropy over clusters.
- P_k: Proportion of candidates in cluster k.
- $\log(m+1)$: Normalization factor to bound NSN between [0,1].

Interpretation: NSN measures semantic consistency. If responses are highly similar (few large clusters), entropy is low and NSN is high, indicating confidence.

Example: Say $m = 6$ responses fall into 3 clusters:

- Cluster A: 3 responses $\rightarrow P_A = 0.50$
- Cluster B: 2 responses $\rightarrow P_B 0.33$
- Cluster C: 1 response $\rightarrow P_C 0.17$

Compute entropy:
We use the Shannon entropy formula over the cluster distribution:

$$H = -\sum_{k=1}^{K} P_k \log P_k$$

Where:

- K: number of clusters
- P_k: proportion of candidate responses in cluster k

Given:

$$P_1 = 0.50, \quad P_2 = 0.33, \quad P_3 = 0.17$$

$$H = -\left(0.50 \log 0.50 + 0.33 \log 0.33 + 0.17 \log 0.17\right) \approx 1.47$$

Normalize:
We normalize by the log of $(m + 1)$ to ensure the NSN score is in $[0, 1]$:

$$\text{NSN}(y_i) = 1 - \frac{H}{\log(m + 1)}$$

Here, $m = 6$ (number of responses), so:

$$\log(6 + 1) = \log 7 \approx 1.95$$

Final result:

$$\text{NSN} = 1 - \frac{1.47}{1.95} \approx 0.25$$

Limitation:

- Quality depends on clustering method and chosen thresholds.
- High NSN may reflect consistency, not correctness.
- Rare but important semantic variants may be underrepresented.

4. BERTScore ($\mathbf{F_1}$) – with Token-level Calculation Example

Formula:

$$\text{BERTF}_1 = \frac{2 \cdot \text{BERTP} \cdot \text{BERTR}}{\text{BERTP} + \text{BERTR}}$$

Notation:

- BERTP: Precision — average of maximum cosine similarities from **each candidate token** to any **reference token**.
- BERTR: Recall — average of maximum cosine similarities from **each reference token** to any **candidate token**.
- Token embeddings from pre-trained models (e.g., BERT/RoBERTa).
- Optional IDF weighting emphasizes rare tokens. (This follows the original BERTScore definition.) :contentReference[oaicite:1]index=1

Token-level Calculation Example:
Assume:

Reference: "The Eiffel Tower" Candidate: "Eiffel Tower is high"

Candidate Token	Cosine with [The, Eiffel, Tower]			Max
Eiffel	0.10	0.92	0.88	0.92
Tower	0.05	0.85	0.95	0.95
is	0.01	0.10	0.12	0.12
high	0.00	0.05	0.08	0.08

$$\text{BERTP} = \frac{0.92 + 0.95 + 0.12 + 0.08}{4} = 0.52$$

Reference Token	Cosine with [Eiffel, Tower, is, high]			Max
The	0.20	0.05	0.01	0.00	0.20
Eiffel	0.92	0.85	0.10	0.05	0.92
Tower	0.88	0.95	0.12	0.08	0.95

$$\text{BERTR} = \frac{0.20 + 0.92 + 0.95}{3} = 0.69$$

Therefore,

$$\text{BERTF}_1 = \frac{2 \cdot 0.52 \cdot 0.69}{0.52 + 0.69} \approx 0.59$$

5. BLEURT (Bilingual Evaluation Understudy with Representations from Transformers)

Overview: BLEURT is a learned evaluation metric that uses a pre-trained and fine-tuned BERT-like model to predict human judgments of text quality. It accounts for fluency, grammar, meaning preservation, and factual consistency.

Key Features:

- Fine-tuned on human-annotated sentence pairs (e.g., WMT, semantic similarity, QA).
- Incorporates contextual embeddings, capturing paraphrasing, reordering, and factuality.
- Outputs a continuous score typically ranging from -1 to 1 (higher = better).

Interpretation: BLEURT is robust to surface-level variations and penalizes hallucinations or factual errors more strongly than BLEU or ROUGE. It correlates well with human preference scores in generation tasks.

Example:

- **Reference:** `"Mars is the fourth planet from the sun."`
- **Candidate A (correct):** `"Mars is the fourth planet from the sun."` $\rightarrow$ BLEURT ≈ 0.98
- **Candidate B (hallucinated):** `"Mars is a gas giant and the second planet."` $\rightarrow$ BLEURT ≈ 0.20

Strengths:

- Captures fluency, paraphrase, factual correctness, and coherence.
- Pretrained on real-world corpora and tuned with task-specific data.
- Better correlation with human judgments than n-gram metrics (e.g., BLEU, ROUGE).

Limitations:

- Requires GPU inference — slower than simpler metrics.
- Sensitive to domain mismatch between training and test data.
- May still miss subtle factual hallucinations if not covered in training.

Reference: Sellam et al., "BLEURT: Learning Robust Metrics for Text Generation," *ACL 2020*. [arXiv:2004.04696](https://arxiv.org/abs/2004.04696)

6. Normalized Cosine Similarity (NCS)

Formula:

$$\text{NCS}(y_i) = \frac{1}{m}\sum_{j=1}^{m}\frac{V(y_i)\cdot V(\tilde{y}_{ij})}{\|V(y_i)\|\|V(\tilde{y}_{ij})\|}, \quad \text{then normalized as: } \frac{\text{sim}+1}{2}$$

Notation:

- y_i: Ground-truth reference sentence.
- $\tilde{y}_{ij}$: j-th candidate response for y_i.
- $V(y)$: Sentence-level embedding vector for sentence y, e.g., from Sentence-BERT, Universal Sentence Encoder.
- m: Number of candidate responses.
- Cosine similarity range: $[-1, 1]$, normalized to $[0, 1]$ for NCS.

Interpretation: NCS evaluates the **semantic similarity** between reference and candidate responses at the **sentence level**, using cosine similarity of embedding vectors. The normalization ensures that the score lies in $[0, 1]$, where 1 indicates maximum similarity.

Example:

- **Reference:** `"The Earth orbits the Sun."`
- **Candidate A:** `"The planet Earth revolves around the Sun."` $\rightarrow$ Cosine $\approx 0.86 \rightarrow$ NCS $= (0.86 + 1)/2 = 0.93$
- **Candidate B:** `"The moon is Earth's satellite."` $\rightarrow$ Cosine $\approx 0.15 \rightarrow$ NCS $= (0.15{+}1)/2 = 0.575$

Strengths:

- Fast and simple to compute with modern sentence encoders.
- Captures overall semantic similarity beyond word overlap.
- Useful for quick black-box LLM uncertainty estimation.

Limitations:

- Sensitive to choice of encoder; results vary across models.
- Sentence-level embedding may overlook token-level nuance.
- Cannot distinguish between factually incorrect but semantically close outputs.

—

BLEU and ROUGE: Definitions and Limitations

BLEU (Bilingual Evaluation Understudy)

Formula:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Notation:

- p_n: Modified n-gram precision for n-grams of length n.
- w_n: Weight for each n-gram (commonly uniform, e.g., $1/4$ for $n = 1$ to 4).
- BP: Brevity penalty to discourage short outputs:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

where c is candidate length, r is reference length.

Interpretation: BLEU measures surface-form overlap via n-gram precision. Higher scores indicate closer word-level matching with the reference.

Example:

- **Reference:** `"The cat sat on the mat."`
- **Candidate B:** `"The feline rested on the mat."` → BLEU ≈ 0.30 (low lexical overlap)
- **Candidate C:** `"The cat sat on the mat and looked outside."` → BLEU ≈ 0.60 (good overlap, longer than reference)

Limitations:

- Fails to capture paraphrasing or semantic similarity.
- Penalizes valid longer generations due to brevity penalty.
- Not sensitive to meaning — high BLEU does not guarantee factual correctness.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Variants:

- **ROUGE-N:** Recall of overlapping n-grams.
- **ROUGE-L:** Longest Common Subsequence (LCS) based recall.
- **ROUGE-S:** Skip-bigram co-occurrence.

Interpretation: ROUGE measures how much of the reference is "recalled" in the candidate. Often used in summarization evaluation.

Example:

- **Reference:** `"The capital of Germany is Berlin."`
- **Candidate A:** `"Berlin is the capital of Germany."` → ROUGE-L ≈ 1.0 (perfect LCS match despite reordering)
- **Candidate B:** `"The capital of Germany is Berlin and France is Paris."` → ROUGE-L ≈ 1.0 (hallucination not penalized)

Limitations:

- Ignores word meaning — only string-level matches count.
- Cannot detect hallucinated or factually wrong information.
- Recall-only variants may reward verbosity or repetition.

Black-Box Scorers Summary Table (Revised)

Scorer	Paraphrase Tolerant	Needs NLI	Speed	Typical Use
EMR	No	No	Very Fast	Closed QA
NCP	Yes	Yes	Medium	Factual QA
NSN	Yes	Yes	Medium	General QA
BERTScore	Strong	No	Slow	Text Generation
BLEURT	Strong	No	Slow	High-Quality Eval
NCS	Moderate	No	Fast	Quick Screening

Table 1: Comparison of black-box UQ scoring methods

Analogy Map

Scorer	Analogy
EMR	Exact same sentence
NCP	No contradiction among versions
NSN	Semantic clustering strength
BERTScore	Do meanings align?
BLEURT	Would a human say these match?
NCS	Are vectors close in semantic space?
BLEU	Did you copy exact n-grams?
ROUGE	Did you recall original phrasing?

—

Key Q&A

- Q: What is an NLI model, and how is it used to assess contradiction between two responses?
  A: It predicts whether one sentence entails, contradicts, or is neutral to another. For hallucination detection, we compute contradiction scores both ways (Ref $\rightarrow$ Cand and Cand $\rightarrow$ Ref).

- Q: Why normalize NSN using $\log(m+1)$?
  A: It rescales entropy to the [0,1] range and avoids undefined values when $m = 0$.

- Q: Why add 1 and divide by 2 in cosine similarity?
  A: To normalize the $[-1, 1]$ range to $[0, 1]$ for consistent scoring. Without it, negative scores would distort averages.

- Q: What does high BERTP but low BERTR mean?
  A: The candidate covers the reference well but adds extra or irrelevant content.

- Q: Why can cosine similarity be negative?
  A: It indicates opposite directions in embedding space — semantically dissimilar or contradictory.

- Q: What happens if token order differs but words match in BERTScore?
  A: BERTScore is largely order-invariant; scores stay high if semantic content is preserved.

- Q: Why can't BLEU or ROUGE detect hallucination reliably?
  A: They match surface n-grams, not meaning, and can't penalize made-up information.

- Q: When can ROUGE be high despite hallucination?
  A: When the candidate matches the reference but appends hallucinated facts.

- Q: What would BLEU say if hallucinated info is added to a perfect copy?
  A: BLEU stays high due to strong n-gram overlap; hallucination goes undetected.

- Q: Give a hallucination example with high BLEU or ROUGE.
  A: Ref: `"Einstein developed the theory of relativity"`
  Candidate: `"Einstein developed the theory of relativity and designed the atomic bomb"`
  ⇒ high BLEU/ROUGE, but hallucinated.

White-Box UQ Metrics for LLM Confidence Estimation

1. Token-Level Entropy

Definition:

$$H_t = -\sum_{w \in V} p_t(w) \log p_t(w)$$

Notation:

- H_t: Entropy at decoding step t.
- V: Vocabulary set.
- $p_t(w)$: Model's predicted probability of token w at time step t.

Description: Token-level entropy quantifies how uncertain the model is when choosing the next token. It reflects the distributional spread over the vocabulary at a specific decoding position.

Interpretation:

- **High** $H_t \rightarrow$ flat probability distribution $\rightarrow$ model is *unsure*.
- **Low** $H_t \rightarrow$ peaked distribution $\rightarrow$ model is *confident*.

Example: Assume the model's predicted probabilities for the top 4 tokens at step t are:

$$\text{Probs} = [0.6,\ 0.2,\ 0.1,\ 0.1]$$

Then the entropy is:

$$H_t = -\left(0.6 \log 0.6 + 0.2 \log 0.2 + 0.1 \log 0.1 + 0.1 \log 0.1\right) \approx 1.57$$

Limitation:

- Token entropy is a **local measure** — it reflects uncertainty at one step only.
- It does **not capture uncertainty at the sequence level** (e.g., overall meaning, hallucination).

2. Sequence Log Probability

Definition:

$$\log P(y) = \sum_{t=1}^{T} \log p_t(y_t)$$

Notation:

- $y = (y_1, \ldots, y_T)$: Generated sequence of length T
- $p_t(y_t)$: Model's predicted probability of token y_t at time step t
- $\log P(y)$: Total log-probability of the generated sequence

Description: Measures the model's overall confidence in the generated sequence, computed by summing the log-probabilities of each token at its respective time step.

Normalized Version:

$$\frac{1}{T} \log P(y)$$

This accounts for sequence length — useful when comparing across outputs of varying lengths.

Example: Given predicted token probabilities:

$$[0.9,\ 0.8,\ 0.7,\ 0.6,\ 0.4]$$

Then:

$$\log P(y) = \log(0.9) + \log(0.8) + \log(0.7) + \log(0.6) + \log(0.4)$$
$$\approx -0.105 - 0.223 - 0.357 - 0.511 - 0.916 = -2.112$$

Normalized:

$$\frac{1}{5} \log P(y) \approx -0.422$$

Limitation:

- A high log-probability reflects fluency and internal confidence — but does **not guarantee factual accuracy**.
- Sequences with hallucinations or factual errors may still have high log-probs if they are linguistically likely.

3. Cumulative Entropy

Definition:

$$\text{CE}(y) = \sum_{t=1}^{T} H_t, \quad \text{or} \quad \text{AvgEnt}(y) = \frac{1}{T} \sum_{t=1}^{T} H_t$$

Notation:

- H_t: Token-level entropy at time step t
- T: Total number of tokens in the generated sequence
- $\text{CE}(y)$: Cumulative entropy — total uncertainty across the sequence
- $\text{AvgEnt}(y)$: Average entropy — normalized per-token uncertainty

Description: Cumulative Entropy (CE) quantifies the total uncertainty the model experiences while generating the entire output sequence. Average Entropy normalizes this by sequence length to enable cross-sequence comparison.

Interpretation:

- High CE $\rightarrow$ the model was uncertain throughout the generation.
- Low CE $\rightarrow$ the model was consistently confident across tokens.

Example: Given token-level entropies:

$$[0.1,\ 0.2,\ 0.3,\ 0.3,\ 0.5,\ 1.0]$$

Then:

$$\text{CE} = 0.1 + 0.2 + 0.3 + 0.3 + 0.5 + 1.0 = 2.4, \quad \text{AvgEnt} = \frac{2.4}{6} = 0.4$$

Limitation:

- CE and AvgEnt measure distributional uncertainty, not semantic accuracy.
- A model can produce fluent yet factually incorrect output with low entropy.

4. Top-k Margin (Prediction Gap)

Definition:

$$\Delta_t = p_1 - p_2 \quad \text{or} \quad \Delta_t = \text{logit}_1 - \text{logit}_2$$

Notation:

- p_1, p_2: Top-1 and Top-2 token probabilities at decoding step t
- logit_1, logit_2: Corresponding logits (optional variant)
- Δ_t: Margin (or "confidence gap") at time step t

Description: Top-k margin quantifies how strongly the model prefers its most likely prediction over the runner-up at each decoding step.

Average Margin:

$$\text{AvgMargin}(y) = \frac{1}{T}\sum_{t=1}^{T} \Delta_t$$

This aggregates the margin over the entire sequence to reflect overall prediction confidence.

Example: At decoding step t, suppose:

$$\text{Top-2 probabilities} = [0.45,\ 0.44]$$

Then:

$$\Delta_t = 0.45 - 0.44 = 0.01 \quad \Rightarrow \quad \text{High uncertainty}$$

Interpretation:

- **Low** $\Delta_t \rightarrow$ the model is indecisive between top predictions.
- **High** $\Delta_t \rightarrow$ the model has strong confidence in its choice.

Strengths:

- Sensitive to fine-grained token-level uncertainty, especially when Top-2 are close.
- Can complement entropy, especially in near-tie cases where entropy appears moderate.

Limitation:

- Focuses only on Top-2 tokens — ignores distributional uncertainty beyond them.
- May not detect hallucinations if top prediction is confidently wrong.

White-Box Scorers Summary Table

Scorer	Granularity	Interpretation	Lower Score = Higher Confidence?
Token Entropy	Token-level	Local uncertainty	Yes
Top-k Margin	Token-level	Certainty gap in logits	No
Log-Probability	Sequence-level	Overall likelihood	No
Cumulative Entropy	Sequence-level	Total generation entropy	Yes

Table 2: Comparison of white-box UQ scoring methods

LLM-as-Judge Confidence Check

Definition: A black-box uncertainty estimation technique where the LLM is asked to evaluate its own answers via natural language prompting. It does not rely on token probabilities, entropy, or semantic similarity, but instead treats the LLM as a self-reflective reasoning agent.

1. Reflexive Prompting (Self-Rated Confidence)

Prompt Template:

> "On a scale from 0 to 100, how confident are you in the above answer? Please explain your rating."

Definition: A black-box self-evaluation technique where the LLM is prompted to assign a numerical confidence score to its own output and provide a justification. Rather than relying on internal token probabilities or external semantic comparisons, this method elicits explicit, self-reported certainty — offering a form of introspective judgment. Recent research suggests that such elicitation often produces more calibrated uncertainty estimates.

Purpose: To capture the model's self-assessed confidence and accompanying reasoning, enabling the detection of hallucinations, miscalibrations, and overconfident errors.

Output Example:

> "Confidence: 90/100. The answer is based on well-known facts and aligns with common knowledge."

Interpretation:

- **High score + vague explanation** → possible overconfidence or hallucination.
- **Low score + specific rationale** → genuine epistemic uncertainty, even if surface fluency is high.

Use Cases:

- Detecting **overconfident hallucinations** — when the model outputs incorrect content with unjustified certainty.
- Identifying **underconfident but correct answers** — especially when white-box metrics (e.g., low entropy) suggest false confidence.

Strengths and Limitations:

- **Strength:** Produces interpretable, human-readable confidence estimates often aligned with factual accuracy.
- **Limitation:** May inherit human-like overconfidence or hedging behaviors; justification quality varies with prompt phrasing and model architecture.
- **Limitation:** Adds computational overhead and may be infeasible in API-only or latency-constrained settings.

2. Chain-of-Verification (CoVe)

Prompt Template:

> "Can you verify your answer step-by-step and explain the reasoning that led to your conclusion?"

Definition: Chain-of-Verification (CoVe) is a self-verification prompting strategy in which the LLM not only produces an initial response but also explicitly verifies it through a structured series of fact-checking steps. This involves generating and answering auxiliary questions to assess the validity of key claims before producing a final, refined response.

Purpose: To detect and correct factual errors, unsupported claims, or hallucinations by decomposing the response into verifiable components and enforcing step-wise validation.

Workflow (4-Step Pipeline):

1. **Initial Response:** The LLM generates a first-pass answer to the user query.
2. **Verification Planning:** The model identifies potentially uncertain or factual elements and formulates verification questions.

3. **Fact-Checking Execution:** Each question is answered separately, using internal knowledge or retrieval to validate specific claims.
4. **Final Answer Refinement:** The original response is revised using verified facts to improve factuality and reduce hallucinations.

Example:

> "Step 1: Identify the subject. Step 2: Recall relevant facts. Step 3: Check each factual statement (e.g., birthplace, dates). Final Answer: Verified."

Strengths:

- Effectively reduces hallucinations in factual QA, summarization, and list-generation tasks.
- Encourages factual rigor by shifting focus from linguistic fluency to claim validation.
- Enables better interpretability through explicit reasoning steps.

Limitations:

- Introduces latency due to multi-step reasoning and generation.
- May fail if the model lacks the capacity to recognize its own errors or generates flawed verification steps.
- Performance is prompt-sensitive and may vary across domains and question types.

3. Retrieval-Augmented Self-Check

Prompt Template:

> "Based on the following retrieved evidence, is your answer supported or contradicted? Please explain."

Definition: Retrieval-Augmented Self-Check is a hybrid black-box technique that combines LLM-based self-evaluation with evidence-based grounding. The model first generates a response, then examines retrieved external documents to assess whether its own output is factually supported or contradicted. This approach builds on Retrieval-Augmented Generation (RAG) and Self-RAG techniques, integrating fact-checking into the reasoning loop.

Purpose: To improve factual reliability by explicitly grounding the LLM's output in external sources, allowing the model to perform post hoc validation and generate evidence-based justifications.

Example:

> "Evidence confirms Barack Obama was born in Hawaii. My answer is supported." "Source: CNN, U.S. Government Records."

Strengths:

- Bridges internal reasoning with external factual verification, reducing reliance on model memory alone.
- Provides human-readable justifications grounded in evidence, aiding transparency and interpretability.
- Empirically shown to reduce hallucinations in long-form QA and summarization tasks.

Limitations:

- Retrieval quality is critical—biased, outdated, or irrelevant evidence may degrade verification accuracy.
- The model may rationalize incorrect outputs using superficially relevant evidence.
- Increases computational overhead due to the retrieval pipeline and additional verification steps.

Method	Output	Advantages	Challenges
Reflexive Prompt	Score + Text	Simple, model introspection	Prompt-sensitive; may mimic human bias
Chain-of-Verification (CoVe)	Rationale	Step-wise fact checking	Verbose; hard to automate evaluation
Retrieval-Augmented Check	Verdict + Evidence	Evidence-grounded; reduces hallucination	Dependent on retrieval quality; adds latency

Table 3: Summary of LLM-as-a-Judge confidence estimation methods

Hybrid Uncertainty Quantification (UQ)

Definition: Hybrid UQ integrates diverse uncertainty signals — from internal model behavior to semantic agreement and self-reflection — to produce a unified and robust estimate of response reliability, particularly useful in hallucination detection. It integrates:

- **White-box scores**: Internal model confidence signals (e.g., entropy, log-probabilities)
- **Black-box scores**: Semantic agreement among sampled outputs
- **LLM-as-Judge scores**: Self-reflection and explanation prompts from the model itself

1. White-Box Signals

- **Token Entropy**: Local uncertainty for each generated token.
- **Top-k Margin**: Gap between top-1 and top-2 logits.
- **Log-Probability**: Overall likelihood assigned to the sequence.
- **Cumulative Entropy**: Total entropy over the sequence.

2. Black-Box Signals

- **EMR (Exact Match Rate)**: Fraction of samples identical to the main response.
- **NCP (Non-Contradiction Probability)**: Semantic consistency using NLI models.
- **NSN (Normalized Semantic Negentropy)**: Cluster tightness of answer variants.
- **BLEURT / BERTScore / NCS**: Semantic similarity to peer outputs.

3. LLM-as-Judge Signals

- **Reflexive Prompting**: Ask the model to self-report confidence.
- **Chain-of-Verification**: Prompt it to verify and re-justify its answer.
- **Retrieval-Aided Justification**: Check its output against external evidence.

4. Hybrid Fusion Formula

$$\text{HybridUQ}(y) = \alpha \cdot \text{WhiteBox}(y) + \beta \cdot \text{BlackBox}(y) + \gamma \cdot \text{LLMJudge}(y)$$

Weight Tuning: Coefficients α, β, and γ control influence of each modality and can be adjusted based on calibration or task.

5. Synthetic Example: "Barack Obama was born in Kenya"

- **WhiteBox Metrics:** Avg Entropy = 0.417 ⇒ Score = 0.583
- **BlackBox Metrics:** EMR = 0.4, NCP = 0.7, BLEURT = 0.83 ⇒ Score = 0.643
- **Judge Score:** Reflexive confidence = 0.7
- **Final Score:** $(0.583 + 0.643 + 0.7)/3 = \mathbf{0.642}$

Interpretation: Although the output appears fluent and plausible, multiple scorers highlight critical weaknesses: high entropy at key tokens, semantic disagreement with alternatives, and low self-confidence. Together, these suggest the answer is a likely hallucination.

Note: To ensure fair contribution across modalities, scores are normalized to [0, 1] before aggregation.

Advantages

- Multi-signal fusion enables robust hallucination detection.
- Captures both surface fluency and deep factual inconsistencies.
- Balances internal signals with semantic behavior and introspection.

Limitations

- Higher computation (sampling + prompting).
- Weight tuning needed for optimal performance.
- Complexity in metric normalization and aggregation.