PROBETOK: REAL-TIME HALLUCINATION DETECTION IN RAG WITH ONE SHARED PROBE TOKEN

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) reduces but does not eliminate hallucination: a response may still contain sentences that the retrieved documents do not support. Such a sentence should be detected in real time since LLMs tend to build on their earlier mistakes. However, most detectors check the complete response and therefore run only after it has been generated, and most of those that can check each sentence in real time need a separate model to read the documents. In this paper, we rethink real-time detection from the perspective of generation and find that, at the moment a sentence is generated, the generator has already read everything a detector needs and holds it in its cache, which makes it a natural basis for a real-time detector. However, its hidden states are computed to generate the next token, not to detect hallucination. Intuitively, asking the generator whether the sentence is supported by the documents would switch it from generation to detection, but asking after every sentence is costly. We therefore propose PROBETOK, which uses a single learned token, the probe token, instead of a handwritten question, and in our experiments the learned token is on average more accurate than the handwritten one. After each sentence is generated, PROBETOK feeds the probe token to the frozen generator. The embedding of the probe token is learned so that the generator computes a hidden state for detection, from which a linear head reads the hallucination score. PROBETOK then removes the probe token and restores the cache, so the generator continues the response as if no check had happened. On the RAGTruth and TRIVIA+ datasets with responses from four of the latest LLMs, PROBETOK achieves the highest average sentence-level AUROC compared with state-of-the-art detectors, and adds only 4.2% to the generation time. Moreover, a strong open LLM can use our method to check the responses of the other generators without access to their hidden states. In our experiments the best such detector is on average only 0.03 AUROC below PROBETOK running on the generator itself.

1 INTRODUCTION

Retrieval-augmented generation (RAG) grounds large language models (LLMs) in retrieved documents to alleviate hallucination (Lewis et al., 2020). However, LLMs may still generate sentences that are not supported by the retrieved documents (Niu et al., 2024). For example, commercial legal research tools built on RAG hallucinate on 17% to 33% of queries (Magesh et al., 2025). In banks, RAG assistants answer customers from policy documents, where an incorrect answer can lead to regulatory and financial risk (Katerenchuk et al., 2026).

Detecting such sentences in real time, as they are generated, is valuable in practice. In particular, LLMs tend to build on their earlier mistakes (Zhang et al., 2024), so an unsupported sentence that goes unnoticed can mislead the sentences that follow. If the sentence is instead detected as soon as it is generated, the system can retrieve more documents and regenerate it (Jiang et al., 2023; Varshney et al., 2023) before the error propagates through the response.

We therefore study *real-time sentence-level hallucination detection* for RAG: after each sentence is generated, a detector decides from the prompt and the response so far whether the sentence is supported. A sentence is the natural unit: unlike a token, it usually makes a complete claim that can be checked, and unlike the complete response, it is available early enough for the check to be timely.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Comparison with state-of-the-art detectors, averaged over four LLMs and two datasets. Lettuce-P and Lettuce-R are the post hoc and real-time versions of LettuceDetect, which score the complete response and each sentence as it is generated, respectively. Extra time is calculated as the detection time divided by the generation time on an L40 GPU. The experiment shows that PROBETOK achieves the highest average sentence-level AUROC at low time cost.

Method	Real time	No extra LM	Extra time	AUROC
TMSF (Itkin, 2026)	✗	✗	1.9%	0.680
MVA (Ogasa & Arase, 2025)	✗	✗	6.5%	0.640
Lettuce-P (Kovács & Recski, 2025)	✗	✗	1.1%	0.732
ReDeEP (Sun et al., 2025)	✗	✓	20.8%	0.613
LUMINA (Yeh et al., 2026)	✗	✓	17.6%	0.634
Lettuce-R (Kovács & Recski, 2025)	✓	✗	7.1%	0.717
MiniCheck (Tang et al., 2024)	✓	✗	18.7%	0.676
Min-Prob (Jiang et al., 2023)	✓	✓	0.0%	0.647
Last-H	✓	✓	0.0%	0.733
ProbeTok	✓	✓	4.2%	**0.763**

Most existing detectors, however, fall short of such cheap real-time detection in two ways. (1) **Post hoc.** Most of them check the complete response. These include external verifiers (Tang et al., 2024; Kovács & Recski, 2025), sampling-based methods (Farquhar et al., 2024) and RAG detectors that read the internal states of the generator (Sun et al., 2025; Yeh et al., 2026). (2) **Extra model.** The verifiers that can check each sentence need a second model that reads the documents in addition to the generator itself. Table 1 summarizes the state-of-the-art detectors that we evaluate, together with Last-H, a linear classifier on the last hidden state of each sentence. Apart from Min-Prob, which is less accurate than the strongest detectors by a wide margin, and Last-H, which we discuss below, none of the existing detectors runs in real time without keeping an extra language model in memory.

In this paper, we first rethink real-time sentence-level hallucination detection from the perspective of generation. At the moment a sentence s_i is generated, a detector needs the retrieved documents D, against which s_i is checked. It also needs the question q and the earlier sentences $s_{<i}$, which determine what s_i claims, since s_i can refer to them through pronouns and ellipsis. Any detector is thus a function $g(q, D, s_{<i}, s_i)$ of these four inputs.

We realize that the generator itself already has this input. **When it predicts the token after s_i, it conditions on exactly the same four inputs, and it has already encoded them in its cache.** The generator is therefore a natural basis for a real-time detector: built on the generator, a detector needs neither a second model nor a second pass over the retrieved documents at every check.

Some detectors already build on the generator and are therefore cheap and real-time. Dynamic RAG methods such as FLARE and DRAGIN (Jiang et al., 2023; Su et al., 2024a) read the token probabilities of the generator to decide when to retrieve, and internal-state detectors (Kossen et al., 2024; Su et al., 2024b) train a classifier on the hidden states of the generator. However, both suffer from a *task mismatch*: the token probabilities and the hidden states are computed to predict the next token, not to detect hallucination. Detectors based on token probabilities, for example, fail when the generator is confidently wrong (Simhi et al., 2025).

Other methods switch the generator from generation to detection by prompting it with a question about its own output. Asked whether a statement is true, an LLM can recognize many of its own mistakes (Kadavath et al., 2022; Zhang et al., 2024), and PRISM (Zhang et al., 2025) reads the hidden states of the generator under such a prompt. However, the question is handwritten rather than learned for the task, and it adds tens of tokens to every check.

Motivated by prompt tuning (Lester et al., 2021), which shows that a few learned input embeddings can replace an instruction for a frozen LLM, and by gist tokens (Mu et al., 2023), which compress a prompt into a few tokens, we propose PROBETOK, which replaces the handwritten question with a single learned token. PROBETOK adds two components to the frozen generator. The first is the *probe token*, whose input embedding is learned so that feeding it after a sentence makes the generator

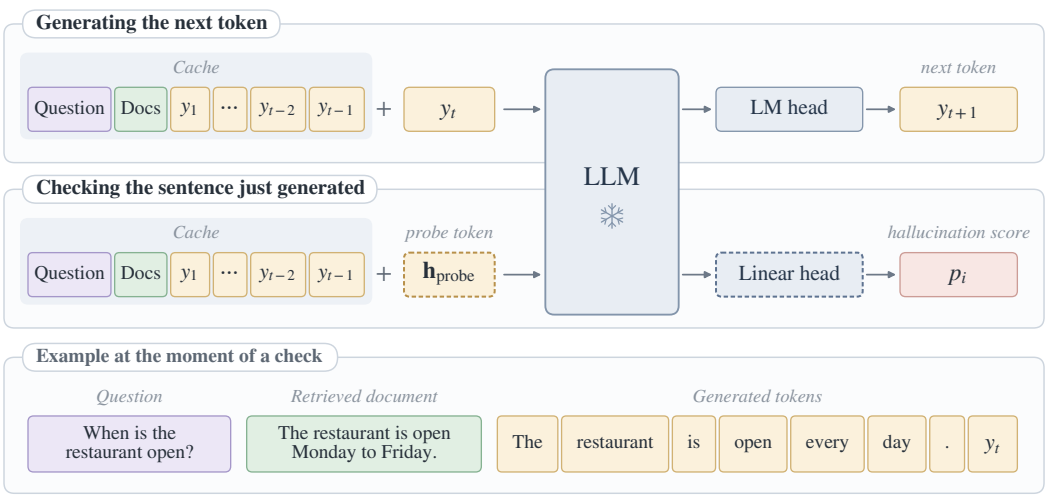


Figure 1: Overview of PROBETOK. Top: the frozen LLM generates the next token from the current token and its cache. Middle: once a sentence has been generated, PROBETOK feeds the probe token in place of the token that follows the sentence, which has been sampled but not yet processed, and a linear head predicts from the resulting hidden state whether the sentence is unsupported. The cache is then restored and generation continues with the next token. The shaded bands mark the input that both rows share and that the cache already holds, and dashed outlines mark the two components that PROBETOK adds. Bottom: an example in which the generated sentence is not supported by the retrieved document. Here y_t is the token generated after the example sentence.

compute a hidden state for detection. The second is a linear head that reads the hallucination score from this state. As illustrated in Figure 1, after each sentence is generated, PROBETOK (1) feeds the probe token to the generator on top of its cache, (2) reads the resulting hidden state with the linear head and (3) removes the probe token and restores the cache, so that generation continues as if no check had happened. Detecting a sentence therefore costs only one extra forward pass, and PROBETOK adds only $2d + 1$ parameters to a generator with hidden size d. The exception is Gemma-4 (Gemma Team, 2026), which uses per-layer embeddings, so our method also learns them for the probe token.

Beyond the detector, we also regenerate the responses of two benchmarks, since existing RAG benchmarks such as RAGTruth (Niu et al., 2024) contain responses from earlier LLMs, for example Llama-2 (Touvron et al., 2023), which no longer reflect the errors of the latest LLMs. We regenerate the responses of RAGTruth and TRIVIA+ with Qwen3.5 (Qwen Team, 2026), Gemma-4 (Gemma Team, 2026), Ministral-3 (Liu et al., 2026) and Granite-4.2 (IBM Granite Team, 2026), all with at most 9B parameters, and label their unsupported sentences with GPT-5.6 SOL (OpenAI, 2026).

Experiments show that PROBETOK is accurate and cheap. Across two datasets and four of the latest LLMs as generators, it achieves the highest average sentence-level AUROC of all detectors, and on TRIVIA+ the highest for all four generators, while adding only 4.2% to the generation time. Our comparison with controls shows that its gain comes from learning the probe token. Specifically, PROBETOK outperforms Last-H on both datasets for all four generators and Fixed probe, which keeps the probe token at its random initialization, in seven of the eight conditions. Since PROBETOK needs only the prompt and the response, it also works in a black-box setting (Sriramanan et al., 2024), in which an open LLM checks the responses of the other generators without access to their hidden states, with a small loss in AUROC.

In summary, our contributions are as follows:

- We propose PROBETOK, a real-time hallucination detector for RAG that checks each sentence with a single learned token instead of a handwritten question, which makes the frozen generator compute a hidden state for detection without changing the response.

- We show on RAGTruth and TRIVIA+ responses regenerated with four of the latest LLMs, which cover full-attention, sliding-window and recurrent architectures, that PROBETOK achieves the highest average sentence-level AUROC of all detectors at small cost.
- Our method can also be applied to an open LLM, which then checks the responses of the other generators in the black-box setting without access to their hidden states.

2 RELATED WORK

Hallucination Detection During Generation. Several methods detect errors during generation. FLARE (Jiang et al., 2023) and DRAGIN (Su et al., 2024a) retrieve more documents when the generated tokens have low probability or high attention-weighted entropy, and Self-RAG (Asai et al., 2024) fine-tunes the generator to produce critique tokens that judge whether each segment is supported. Other methods read the hidden states of the generator during generation: MIND (Su et al., 2024b) trains a classifier on them, and recent probes flag hallucinated entities (Obeso et al., 2025) or reasoning steps (Lu et al., 2026a) in the same pass. RAGognizer (Ridder et al., 2026) fine-tunes the generator with a token-level hallucination detection head, and the hallucination detection intrinsic of Granite (Danilevsky et al., 2025) scores each sentence with a LoRA adapter after generation. Unlike these methods, PROBETOK neither relies on token confidence nor reads the hidden states computed for the next token, but makes the generator compute a hidden state for detection.

Detection from Internal States and Prompted Self-Checks. Many detectors train a classifier on the hidden states of an LLM (Azaria & Mitchell, 2023; Kossen et al., 2024) or on its attention maps (Chuang et al., 2024), or score its hidden states, attention and logits without training (Sriramanan et al., 2024). For RAG, ReDeEP (Sun et al., 2025) and LUMINA (Yeh et al., 2026) measure how much the generator relies on the documents rather than its parametric knowledge, and PRISM (Zhang et al., 2025) reads the hidden states under a prompt generated from a handwritten seed.

Learned Input Embeddings and Cache Reuse. Prompt tuning (Lester et al., 2021) and prefix tuning (Li & Liang, 2021) adapt a frozen LLM by learning continuous prompts that stay in the context, and gist tokens (Mu et al., 2023) train an LLM to compress a prompt into a few tokens, whereas the probe token of PROBETOK is removed after each check. Concurrent work (Mrykhin & Malykh, 2026) learns prompt embeddings that are inserted into the input, by default at its beginning, and probes the hidden states after a complete answer, so with this placement a check cannot reuse the cache of the generation. Other work reuses the generation cache for a check: Activated LoRA (Greenewald et al., 2025) runs adapters on the cache after generation, KV-PRM (Kuang et al., 2026) appends a verify token with LoRA to score reasoning steps, and Speculative Probing (Zhang et al., 2026) appends soft prompts that a frozen draft head reads. PROBETOK differs in that the generator stays frozen, the probe token is added at every sentence boundary during decoding with the cache restored afterwards, and the target is sentence-level hallucination in RAG. Appendix A reviews further work.

3 METHOD

We first define the task, then derive PROBETOK from the observation that the generator already holds the input of a real-time detector, and finally describe training and detection during generation.

3.1 PROBLEM FORMULATION

Given a prompt $\mathbf{x}$ that contains an instruction, the retrieved documents D and, for question answering, a question q, a frozen LLM f_θ generates a response $\mathbf{y} = (y_1, \ldots, y_T)$. We split the response into sentences $s_1, \ldots, s_N$ with a rule-based splitter and denote by n_i the position of the last token of s_i. Each sentence has a label $c_i \in \{0, 1\}$, where $c_i = 1$ means that s_i contains content that is not supported by the retrieved documents. As in RAGTruth (Niu et al., 2024), support is judged against the retrieved documents rather than against world knowledge, so a true sentence that the documents do not state also counts as unsupported. Real-time detection predicts c_i from $\mathbf{x}$ and $y_{1:n_i}$ alone as soon as s_i has been generated, and it must be cheap enough to run after every sentence.

3.2 Shared Input of the Generator and the Detector

We model a real-time detector as an estimate of the probability that s_i is unsupported given everything that is available at the moment at which s_i has been generated,

$$p_i \approx P(c_i = 1 \mid \mathbf{x}, y_{1:n_i}) = P(c_i = 1 \mid q, D, s_{<i}, s_i), \tag{1}$$

where $s_{<i}$ denotes the earlier sentences and the instruction in $\mathbf{x}$ is fixed. The documents D are the evidence against which s_i is checked, and q and $s_{<i}$ determine what s_i claims. The generator, in turn, predicts the token after s_i from

$$P(y_{n_i+1} \mid \mathbf{x}, y_{1:n_i}) = \mathrm{softmax}\big(\mathbf{W}_{\mathrm{LM}}\mathbf{h}_{n_i}\big), \qquad \mathbf{h}_{n_i} = f_\theta\big(\mathbf{e}(y_{n_i}); \mathcal{C}_{n_i-1}\big), \tag{2}$$

where $\mathcal{C}_n$ denotes the cache of f_θ after it has processed $[\mathbf{x}, y_{1:n}]$, $\mathbf{e}(\cdot)$ is the token embedding of f_θ, $f_\theta(\mathbf{a}; \mathcal{C})$ is the last-layer hidden state that f_θ computes for the input embedding $\mathbf{a}$ with cache $\mathcal{C}$, and $\mathbf{W}_{\mathrm{LM}}$ is the language-model head. Equations 1 and 2 condition on the same input and differ only in what they predict. The generator has therefore already read everything the detector needs, and its cache $\mathcal{C}_{n_i}$ encodes this input. A detector built on the generator needs neither a second model nor a second pass over $\mathbf{x}$, because a forward step of f_θ on top of $\mathcal{C}_{n_i}$ reads the same encoding of the input that the generator reads for the next token. Work on safety monitors has made the analogous observation for policy models (Cunningham et al., 2025; Xuan et al., 2025).

There are two ways to build such a detector. The first reads what the generator has already computed, namely the hidden state $\mathbf{h}_{n_i}$, as existing internal-state detectors do (Azaria & Mitchell, 2023; Kossen et al., 2024). We call this readout Last-H:

$$p_i^{\text{Last-H}} = \sigma\big(\mathbf{w}^\top \bar{\mathbf{h}}_{n_i} + b\big), \tag{3}$$

where σ is the sigmoid function, $\bar{\mathbf{h}}_{n_i}$ is $\mathbf{h}_{n_i}$ standardized with a mean and a standard deviation computed on training sentences, and $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the parameters of a linear head, with d the hidden size of f_θ. However, $\mathbf{h}_{n_i}$ is computed for Equation 2, not for Equation 1: it is a d-dimensional summary of the context shaped by next-token prediction, and a classifier on it can use only what this summary keeps. The second way, which PROBETOK takes, makes the generator compute a state for Equation 1. One more input position, fed on top of $\mathcal{C}_{n_i}$, passes through all layers of f_θ and attends to what the cache has kept of the question, the documents and the sentence, so it is not limited to the d numbers that $\mathbf{h}_{n_i}$ keeps, provided that its input is trained for detection.

3.3 Scoring a Sentence with a Probe Token

After sentence s_i is generated, PROBETOK feeds a learned embedding $\mathbf{h}_{\mathrm{probe}} \in \mathbb{R}^d$ to f_θ and applies a linear head to the resulting hidden state:

$$\mathbf{u}_i = f_\theta(\mathbf{h}_{\mathrm{probe}}; \mathcal{C}_{n_i}), \qquad p_i = \sigma\big(\mathbf{w}^\top \bar{\mathbf{u}}_i + b\big), \tag{4}$$

where $\bar{\mathbf{u}}_i$ is $\mathbf{u}_i$ standardized as in Equation 3. We call $\mathbf{h}_{\mathrm{probe}}$ the probe token and use the same probe token and head for every sentence. Because $\mathbf{h}_{\mathrm{probe}}$ is trained with the detection loss in Equation 5, $\mathbf{u}_i$ is computed for detection, whereas Last-H reads a state computed for generation. The two also differ in position, since PROBETOK reads the state one position later, which the Fixed probe control in Section 4 accounts for. The head is linear in both cases, but the mapping from $\mathbf{h}_{\mathrm{probe}}$ to $\mathbf{u}_i$ passes through the nonlinear layers of the frozen LLM, and the head is trained for detection jointly with $\mathbf{h}_{\mathrm{probe}}$ on the labeled sentences of each condition.

3.4 Training

We train the probe token and the head, $\phi = (\mathbf{h}_{\mathrm{probe}}, \mathbf{w}, b)$, on labeled sentences and keep all parameters of f_θ frozen. The loss is the class-weighted binary cross-entropy

$$\mathcal{L}(\phi) = -\sum_i \big[\alpha\, c_i \log p_i + (1 - c_i) \log(1 - p_i)\big], \tag{5}$$

where α is the ratio of supported to unsupported sentences in the training set, which gives the rare unsupported sentences the same total weight as the supported ones. The class weighting shifts p_i away from the posterior in Equation 1, so we treat p_i as a score and select every threshold on the

validation set. During training, we feed each recorded response to f_θ sentence by sentence and apply the probe token after every sentence. Apart from Gemma-4, the only trainable parameters are the d entries of the probe token and the $d + 1$ parameters of the head. The gradients reach the probe token through the frozen LLM. Gemma-4 also feeds a per-layer embedding of each token to its layers, so we additionally learn a 42×256 per-layer embedding of the probe token, and Appendix C describes the initialization, the optimizer and the selection of the configuration.

3.5 Detection During Generation

Algorithm 1 in Appendix C summarizes how ProbeTok runs during generation. Processing the probe token adds an entry to the cache and can overwrite parts of it, so ProbeTok undoes its effect after the check: it removes the appended key and value of a full-attention layer, puts back the entry that a sliding-window layer (Beltagy et al., 2020) evicts, and restores the recurrent state that a layer such as Gated DeltaNet (Yang et al., 2025) overwrites, which it saves before the check. Appendix C details this for the four LLMs in our experiments, which cover all three cases.

Each check costs one forward pass on a single token, and the exact restoration of the cache leaves the response identical to the one generated without detection.

This is the white-box setting of LLM-Check (Sriramanan et al., 2024), in which the generator is available for detection. In the black-box setting, an open LLM serves as the detector and scores the responses of the generator with its own probe token and head, as Section 4.6 evaluates.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on RAGTruth (Niu et al., 2024), which covers question answering, summarization and data-to-text writing, and on TRIVIA+ (Chen et al., 2026), which contains questions from five sources, including TriviaQA (Joshi et al., 2017) and DROP (Dua et al., 2019). We keep their prompts and retrieved documents and generate new responses with Qwen3.5-9B, Gemma-4-E4B, Ministral-3-8B and Granite-4.2-8B, which we call Qwen, Gemma, Ministral and Granite. Each dataset–generator pair forms one condition, which gives eight conditions. Following an explicit annotation protocol, GPT-5.6 SOL annotates the unsupported spans of every response or abstains, in which case the response is excluded. A sentence is labeled unsupported if it overlaps an unsupported span. A human reviewer labeled 71 sampled sentences and initially disagreed on 14, but in 12 of these the reviewer accepted the label after seeing the judgment of the annotator, a check that Chuang et al. (2024) also applied to their LLM annotator, so the final agreement is 69 of 71. Appendix B describes the splits, the annotation and the review, and Table A1 gives the statistics of each condition.

Compared Detectors. We compare ProbeTok with six state-of-the-art detectors. Lettuce (Kovács & Recski, 2025), MiniCheck (Tang et al., 2024), TMSF (Itkin, 2026) and MVA (Ogasa & Arase, 2025) run a separate model, whereas ReDeEP (Sun et al., 2025) and LUMINA (Yeh et al., 2026) compute features from the internal states of the generator. We retrain TMSF and MVA on the training set of each condition with three seeds, as for ProbeTok, and use the released models and implementations of the other four detectors, which were trained before our response sets existed, for example Lettuce on the original RAGTruth responses. Lettuce-P scores the complete response post hoc, whereas Lettuce-R scores each sentence in real time from the response so far, and MiniCheck also runs in real time. We also evaluate Min-Prob, the retrieval trigger of FLARE, which scores a sentence by one minus the lowest probability of its tokens. Appendix D gives the configurations of all compared detectors. We train ProbeTok separately for each condition with AdamW (Loshchilov & Hutter, 2019), select its configuration on the validation set, as Appendix C describes, and report the mean over three seeds for ProbeTok, TMSF and MVA, the three detectors that are trained on our data.

Controls. To find where the gain of ProbeTok comes from, we evaluate four controls on the same data, in the spirit of control tasks for probes (Hewitt & Liang, 2019). Last-H applies a linear head to the last hidden state of the sentence, as in Equation 3. Fixed probe feeds the probe token at its random initialization and trains only the head. Two further controls insert, after each sentence, a handwritten question that asks the generator whether the last sentence of its response contains a

Table 2: Sentence- and response-level AUROC and F1 of hallucination detection methods on RAGTruth and TRIVIA+ for the responses of four LLMs. PROBETOK has the highest sentence-level AUROC in six of the eight conditions. AUC denotes AUROC, and AUROC and F1 are multiplied by 100. Real time marks the methods that score each sentence as soon as it is generated. PROBETOK, TMSF and MVA are averaged over three seeds. The best results for each generator, level and metric are highlighted in bold, and Table A5 gives paired intervals.

RAGTruth																
Method	Real time	Qwen3.5-9B				Gemma-4-E4B				Ministral-3-8B				Granite-4.2-8B			
		Sentence		*Response*		*Sentence*		*Response*		*Sentence*		*Response*		*Sentence*		*Response*	
		AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
TMSF	✗	64.0	28.9	72.6	**77.5**	69.5	13.4	70.7	37.9	64.8	38.9	**78.5**	84.7	64.0	28.6	83.5	79.6
MVA	✗	60.8	26.1	77.5	72.6	62.0	10.2	56.6	35.4	63.2	35.1	73.2	83.3	56.7	24.7	57.1	71.1
ReDeEP	✗	57.6	26.2	69.6	71.6	56.5	11.2	62.1	33.3	62.0	36.8	71.3	80.8	61.0	28.4	72.3	75.1
LUMINA	✗	63.5	28.9	72.8	74.7	63.5	12.1	71.4	**45.1**	60.8	38.2	75.0	84.0	54.3	25.6	69.5	72.9
Lettuce-P	✗	71.7	34.9	71.0	72.1	**72.3**	11.0	69.1	41.3	**75.9**	**48.3**	74.3	81.7	72.1	35.4	78.6	75.7
Lettuce-R	✓	69.7	34.6	71.1	73.2	70.0	**15.3**	68.7	44.0	75.0	47.1	74.1	**85.1**	72.3	35.7	79.0	75.8
MiniCheck	✓	70.0	33.6	70.8	72.9	65.9	12.9	67.4	41.1	70.5	43.2	76.3	84.0	71.3	33.9	80.5	77.9
ProbeTok	✓	**73.1**	**36.2**	**78.9**	75.8	72.1	**15.3**	**71.9**	44.7	68.4	40.6	69.1	82.7	**74.6**	**36.7**	**84.7**	**79.9**
TRIVIA+																	
TMSF	✗	73.0	31.9	78.5	**62.8**	70.5	20.2	62.4	25.8	69.8	55.6	83.5	79.9	68.4	17.2	71.5	56.1
MVA	✗	64.9	22.2	70.1	53.0	64.5	6.5	69.4	20.4	73.9	58.3	81.8	79.4	66.3	15.0	61.6	51.0
ReDeEP	✗	61.9	23.7	67.8	51.2	56.4	10.3	63.7	33.6	68.2	54.8	80.7	78.7	66.9	16.2	72.0	60.0
LUMINA	✗	74.4	32.6	**79.9**	61.1	62.2	18.1	65.9	36.7	72.2	57.8	86.9	**80.6**	56.1	11.1	74.2	56.8
Lettuce-P	✗	71.1	33.8	68.7	53.1	76.4	26.9	67.0	29.2	77.9	62.2	82.8	76.5	68.1	16.8	65.9	54.5
Lettuce-R	✓	69.5	27.8	68.3	52.8	72.4	21.8	67.4	36.3	77.7	61.7	81.8	79.0	67.2	15.2	70.0	55.1
MiniCheck	✓	64.3	26.9	67.4	53.8	64.7	13.9	73.9	40.4	75.4	59.4	85.2	**80.6**	58.5	10.3	75.4	58.0
ProbeTok	✓	**79.0**	**39.6**	77.8	59.5	**79.8**	**31.3**	**76.9**	**45.1**	**84.2**	**63.8**	**89.2**	80.0	**79.4**	**31.0**	**77.5**	**61.5**

claim that the documents do not support. P(Yes) reads the answer as the probability of Yes among Yes and No without any training, and Question-H trains a linear head on the hidden state at the end of the question, as Last-H does at the end of the sentence. Fixed probe reuses the configuration of PROBETOK, and Last-H and Question-H select their configurations on the validation set and are averaged over three seeds, as PROBETOK is in Table 2.

Evaluation Metrics. We report AUROC and F1 at the sentence level and at the response level. A response is unsupported if any of its sentences is unsupported, and its score is the maximum of its sentence scores. AUROC is our main metric, and F1 uses the threshold selected on the validation set. For each condition, level and metric, we compare PROBETOK with the strongest detector using paired bootstrap intervals over resamples of the test responses (Efron, 1979), as Appendix E describes, and Appendix F reports precision, recall and calibration.

4.2 MAIN RESULTS

PROBETOK is the most accurate detector in most conditions of Table 2, although it sees only the response generated so far while most of the compared detectors see the complete response. PROBETOK achieves the highest sentence-level AUROC in six of the eight conditions, and its average of 0.763 exceeds that of the strongest compared detector, Lettuce-P, which reaches 0.732. The gains are largest on TRIVIA+, where PROBETOK outperforms the strongest detector for all four generators by 0.034 to 0.109 in sentence-level AUROC. On RAGTruth, PROBETOK is the best detector for Qwen and Granite and trails Lettuce-P by 0.002 for Gemma. The paired intervals in Table A5 exclude zero in favor of PROBETOK in three conditions, all on TRIVIA+, exclude zero against it in one, RAGTruth–Ministral, and include zero in the remaining four. Min-Prob reaches only 0.647 on average and falls below PROBETOK in all eight conditions, as Table A4 shows. Part of this lead may

Table 3: Sentence-level AUROC, multiplied by 100, of PROBETOK and the four controls of Section 4, which Appendix C describes. PROBETOK has the highest average. The best result in each column is highlighted in bold, and Table A3 adds the response level.

Method	RAGTruth				TRIVIA+				Avg.
	Qwen	Gemma	Ministral	Granite	Qwen	Gemma	Ministral	Granite	
P(Yes)	52.0	46.1	57.3	49.6	70.2	64.5	63.6	61.7	58.1
Question-H	69.9	**74.6**	**71.7**	71.0	77.1	**81.1**	82.5	70.8	74.8
Fixed probe	64.3	64.0	67.4	63.8	74.8	70.7	81.2	78.4	70.6
Last-H	65.7	69.7	64.7	68.1	78.6	77.0	83.7	78.7	73.3
ProbeTok	**73.1**	72.1	68.4	**74.6**	**79.0**	79.8	**84.2**	**79.4**	**76.3**

come from training on labels of the same condition, which the released detectors have not seen and Last-H shares, so the comparison with Last-H in Section 4.4 isolates the probe token.

RAGTruth–Ministral is the only condition in which PROBETOK clearly trails a detector, with an AUROC of 0.684 against 0.759 for Lettuce-P, whereas on the TRIVIA+ responses of the same generator it reaches 0.842 and outperforms every detector. Appendix H traces the gap to data-to-text, where the errors of Ministral must be checked value by value and suit a token-level verifier, and to summarization. Appendix G compares the detectors within each task, where the clearest gains of PROBETOK are on DROP, and Appendix L reports the results on the unsupported sentences that the generator wrote with high confidence according to its token probabilities.

4.3 EFFICIENCY

Apart from Min-Prob and Last-H, which read what the generator computes anyway, PROBETOK is the cheapest real-time detector. Table 1 reports the time that each detector adds to generation on an L40 GPU, as a percentage of the generation time, measured with batch size one, as Appendix I describes. PROBETOK adds 4.2% on average, against 7.1% for Lettuce-R and 18.7% for MiniCheck, which both encode the documents again for every check. A single check of PROBETOK takes 25 to 37 ms, about as long as generating one token, so a system that holds each sentence until it has been checked delays the response by little more than the time of one token.

4.4 COMPARISON WITH CONTROLS

Table 3 compares PROBETOK with the four controls and shows where its gain comes from. P(Yes) is the weakest detector, with an average AUROC of 0.581. Question-H, which reads the same hidden state with a trained linear head, raises the AUROC in all eight conditions, to 0.748 on average, so the generator does compute a useful state under the question. Replacing the handwritten question with the learned probe token raises the average further to 0.763, while each check costs one input position instead of 29 to 40 and adds 4.2% to the generation time instead of 5.1%, as Appendix I details. The AUROC is higher in five of the eight conditions, and the paired intervals exclude zero in three conditions against one in the other direction, as Table A3 shows. Fixed probe adds the same input position but does not learn its embedding, and it trails PROBETOK in all eight conditions of Table 3. In the paired comparison at the same seed, it trails in seven of the eight, with intervals that exclude zero in six, as Figure A1 and Table A7 show.

Last-H is itself a strong detector, since its average AUROC of 0.733 over three seeds is above that of every compared real-time detector in Table 2. PROBETOK outperforms Last-H in all eight conditions, by 0.031 on average, and Table A6 shows that the paired intervals exclude zero for three of the four RAGTruth generators and for none in the other direction at either level.

4.5 RELIANCE ON THE RETRIEVED DOCUMENTS

PROBETOK uses the retrieved documents in addition to cues in the response itself. To show this, we replace the documents with filler text of the same length and, as a control, the instructions instead. Replacing the documents lowers the sentence-level AUROC of PROBETOK in all eight conditions,

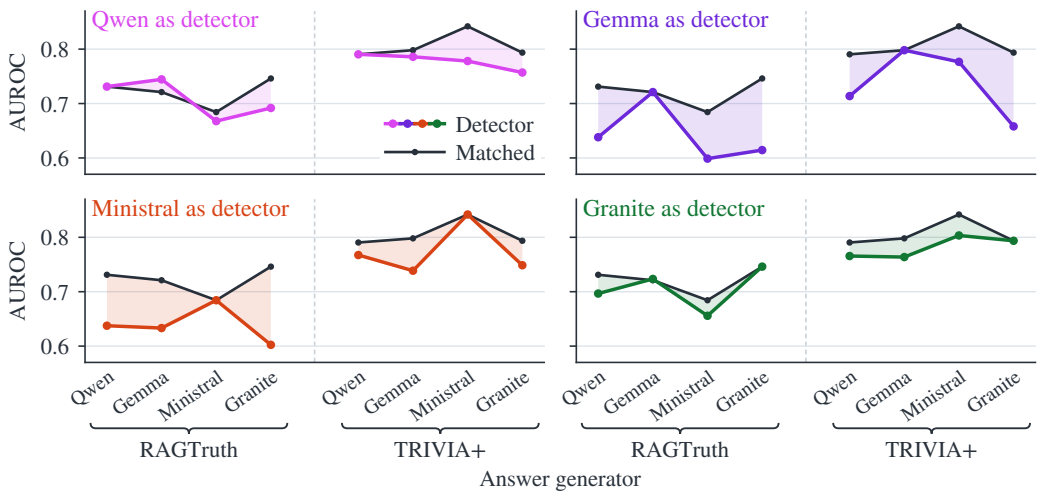


Figure 2: Sentence-level AUROC in the black-box setting, in which PROBETOK runs on one LLM, the detector, to check the responses of each generator. Each panel fixes the detector. Matched, the black line, uses each generator as its own detector and is a reference, not an upper bound. The shaded gap to Matched stays small for Qwen and Granite as detectors. Table A21 lists the values with their standard deviations over the three training seeds of each detector.

by 0.026 to 0.128, more than replacing the instructions does in seven of the eight and more than Last-H loses in six, as Figure A2 shows. Editing only the fact that supports a sentence raises its score, whereas editing an unrelated fact does not, as Appendix J shows.

4.6 BLACK-BOX DETECTION AND TRANSFER ACROSS DATASETS

In the black-box setting, another LLM runs PROBETOK as the detector, with the probe token and the head that were trained on its own responses to the same dataset, so no labels are needed for the responses of a generator whose hidden states are inaccessible. Figure 2 shows the sentence-level AUROC for every detector–generator pair, together with Matched, in which each generator checks itself. Matched is a reference, not an upper bound, since another LLM can be the better detector. Qwen and Granite are the strongest detectors: on the six response sets of the other generators, they average 0.737 and 0.735, respectively, about 0.03 below Matched, and Qwen even outperforms every compared detector on three of these six response sets, as Table A21 lists.

Trained detectors lose accuracy in new domains (Zhang et al., 2025; Deng et al., 2026), and how much depends on the task: a detector trained on TRIVIA+ loses little on RAGTruth question answering, which they share, but much on summarization and data-to-text, as Appendix K reports for each task.

5 CONCLUSION

We presented PROBETOK, a real-time hallucination detector for RAG that checks each sentence with a single learned token instead of a handwritten question. After each sentence, PROBETOK feeds one learned probe token to the frozen generator, reads a hidden state with a linear head and removes the token. On RAGTruth and TRIVIA+ responses from four of the latest LLMs, PROBETOK achieves the highest average sentence-level AUROC of all detectors, including those that read the complete response with a separate model, while adding 4.2% to the generation time. Our experiments also show that a strong open LLM can use it to check the responses of the other three generators on average within 0.03 AUROC of PROBETOK running on the generator itself. These results show that a frozen LLM can check its own sentences as it writes them, at the cost of only one extra forward pass per sentence. Appendix M discusses the limitations of this work and future work.

AI USE STATEMENT

We used generative AI tools for five purposes in this work: research ideation and execution, generation and annotation of the response sets, retrieval and discovery of related work, drafting of sections of the manuscript and revision of the text for clarity and presentation. For the research, the tools helped us discuss hypotheses and experimental designs, implement and debug methods, organize experiments and check statistical analyses. They also helped prepare tables, figures and reproducibility materials from the recorded experimental outputs. For the related work, they helped identify relevant papers, and the authors remain responsible for checking all sources and citations against the original papers.

The response sets contain responses generated by LLMs, and GPT-5.6 SOL marked the unsupported spans from which we derive the sentence labels, as described in Appendix B, and produced the edited documents of the fact-editing analysis in Appendix J. A human review found the label of the annotator correct for 69 of 71 sampled sentences. AI assistance also supported data processing, consistency checks and the interpretation of results. All numbers in this paper come from recorded computations rather than from a writing assistant. The authors are responsible for verifying the AI-assisted text, citations, code, data and analyses, and retain full responsibility for the content and the scientific claims of this paper.

REPRODUCIBILITY STATEMENT

Appendices B to D describe the construction of the data, the detection procedure, training, model selection and the configurations of the compared detectors together with the released models they use. The supplementary material contains the code of PROBETOK with the configurations and a script for all 24 training runs, the data files underlying the main results of PROBETOK and instructions for reproducing the PROBETOK entries of the main table.

ETHICS STATEMENT

PROBETOK detects content that is not supported by the retrieved documents. Neither the generators nor the detector guarantees that a response is factually correct. A missed detection can leave an unsupported statement in a response, and a false alarm can cause a supported statement to be revised or withheld. The use of the benchmarks and of third-party models follows their original terms. PROBETOK is intended to support, not to replace, human review in high-stakes applications.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (Workshop Track)*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJ4-rAVtl.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*. OpenReview.net, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated MAchine reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. URL https://arxiv.org/abs/1611.09268.

Forrest Sheng Bao, Miaoran Li, Renyi Qu, Ge Luo, Erana Wan, Yujia Tang, Weisi Fan, Manveer Singh Tamber, Suleman Kazi, Vivek Sourabh, Mike Qi, Ruixuan Tu, Chenyu Xu, Matthew Gonzales, Ofer Mendelevitch, and Amin Ahmad. FaithBench: A diverse hallucination benchmark for summarization by modern LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 448–461, Albuquerque, New Mexico, apr 2025. Association for Computational Linguistics. ISBN 979-8-89176-190-2. doi: 10.18653/v1/2025.naacl-short.38. URL https://aclanthology.org/2025.naacl-short.38/.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, mar 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7/.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

Bespoke Labs. Bespoke-MiniCheck-7B. https://huggingface.co/bespokelabs/Bespoke-MiniCheck-7B, 2024.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*, pp. 3056–3076, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/0d1986a61e30e5fa408c81216a616e20-Paper-Conference.pdf.

Wenbo Chen, Veena Padmanabhan, Tootiya Giyahchi, Elaine Wong, and Leman Akoglu. Rethinking evaluation for LLM hallucination detection: A desiderata, a new RAG-based benchmark, new insights. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14912–14931, San Diego, California, United States, jul 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.680. URL https://aclanthology.org/2026.acl-long.680/.

Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James R. Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1419–1436, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.84. URL https://aclanthology.org/2024.emnlp-main.84/.

Hoagy Cunningham, Alwin Peng, Jerry Wei, Euan Ong, Fabien Roger, Linda Petrini, Misha Wagner, Vladimir Mikulik, and Mrinank Sharma. Cost-effective constitutional classifiers via representation re-use. Alignment Science Blog, 2025. URL https://alignment.anthropic.com/2025/cheap-monitors/.

Marina Danilevsky, Kristjan Greenewald, Chulaka Gunasekara, Maeda Hanafi, Lihong He, Yannis Katsis, Krishnateja Killamsetty, Yulong Li, Yatin Nandwani, Lucian Popa, Dinesh Raghu, Frederick Reiss, Vraj Shah, Khoi-Nguyen Tran, Huaiyu Zhu, and Luis Lastras. A library of LLM intrinsics for retrieval-augmented generation. arXiv preprint arXiv:2504.11704, 2025. URL https://arxiv.org/abs/2504.11704.

Elizabeth R. DeLong, David M. DeLong, and Daniel L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, 44(3):837–845, 1988. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/2531595.

Yongxin Deng, Zhen Fang, Sharon Li, and Ling Chen. Beyond in-domain detection: SpikeScore for cross-domain hallucination detection. In *The Fourteenth International Conference on Learning Representations*, pp. 115807–115845, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/bc667ac84ef58f2b5022da97a465cbab-Paper-Conference.pdf.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2368–2378, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. ISSN 00905364, 21688966. URL http://www.jstor.org/stable/2958830.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/S41586-024-07421-0. URL https://doi.org/10.1038/s41586-024-07421-0.

Gemma Team. Gemma 4 technical report. arXiv preprint arXiv:2607.02770, 2026. URL https://arxiv.org/abs/2607.02770.

Kristjan Greenewald, Luis Lastras, Thomas Parnell, Vraj Shah, Lucian Popa, Giulio Zizzo, Chulaka Gunasekara, Ambrish Rawat, and David Cox. Activated LoRA: Fine-tuned LLMs for intrinsics. In *Advances in Neural Information Processing Systems*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/4d0b6303d4a4811445f69f357bf6def5-Abstract-Conference.html.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2733–2743, Hong Kong, China, nov 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275/.

Junjie Hu, Gang Tu, ShengYu Cheng, Jinxin Li, Jinting Wang, Rui Chen, Zhilong Zhou, and Dongbo Shan. HARP: Hallucination detection via reasoning subspace projection. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ShEDWasmDG.

IBM Granite Team. Granite 4.2 LLMs: How they're built. https://huggingface.co/blog/ibm-granite/granite-4-2, August 2026.

Igor Itkin. Temporal multi-signal fusion for token-level hallucination detection. arXiv preprint arXiv:2608.18115, 2026. URL https://arxiv.org/abs/2608.18115.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, jul 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Saurav Kadavath, Tom Conerly, Amanda Askell, et al. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022. URL https://arxiv.org/abs/2207.05221.

Hossein Hosseini Kasnavieh, Gholamreza Haffari, Christopher Leckie, and Adel N. Toosi. IntroLM: Introspective language models via prefilling-time self-evaluation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12313–12326, San Diego, California, United States, jul 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.598. URL https://aclanthology.org/2026.findings-acl.598/.

Denys Katerenchuk, Pablo Duboue, and Keelan Evanini. FinRAG-12B: A production-validated recipe for grounded question answering in banking. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 1317–1328, San Diego, California, USA, jul 2026. Association for Computational Linguistics. ISBN 979-8-89176-394-4. doi: 10.18653/v1/2026.acl-industry.92. URL https://aclanthology.org/2026.acl-industry.92/.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in LLMs. *arXiv preprint arXiv:2406.15927*, 2024. doi: 10.48550/arXiv.2406.15927. URL https://arxiv.org/abs/2406.15927.

Ádám Kovács and Gábor Recski. LettuceDetect: A hallucination detection framework for RAG applications. arXiv preprint arXiv:2502.17125, 2025. URL https://arxiv.org/abs/2502.17125.

Peng Kuang, Haibo Jin, Xiaoyu Han, Yanli Wang, Xiaopeng Yuan, Ye Yu, Kaidi Xu, and Haohan Wang. KV-PRM: Efficient process reward modeling via KV-cache transfer for multi-agent test-time scaling. arXiv preprint arXiv:2607.09153, 2026. URL https://arxiv.org/abs/2607.09153.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177, 2022. doi: 10.1162/tacl_a_00453. URL https://aclanthology.org/2022.tacl-1.10/.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, nov 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6449–6464, Singapore, dec 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.397. URL https://aclanthology.org/2023.emnlp-main.397/.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41451–41530. Curran Associates, Inc., 2023b. doi: 10.52202/075280-1797. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/81b8390039b7302c909cb769f8b6cd93-Paper-Conference.pdf.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, aug 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026. URL https://arxiv.org/abs/2601.08584.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8/.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Haolang Lu, Minghui Pan, Ripeng Li, Guoshun Nan, Jialin Zhuang, Zijie Zhao, Zhongxiang Sun, Kun Wang, and Yang Liu. Streaming hallucination detection in long chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21157–21183, San Diego, California, United States, jul 2026a. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1064. URL https://aclanthology.org/2026.findings-acl.1064/.

Pengqian Lu, Jie Lu, Anjin Liu, and Guangquan Zhang. TPA: Next token probability attribution for detecting hallucinations in RAG. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25273–25292, San Diego, California, United States, jul 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1159. URL https://aclanthology.org/2026.acl-long.1159/.

Varun Magesh, Faiz Surani, Matthew Dahl, Mirac Suzgun, Christopher D. Manning, and Daniel E. Ho. Hallucination-free? assessing the reliability of leading AI legal research tools. *Journal of Empirical Legal Studies*, 22(2):216–242, 2025. doi: https://doi.org/10.1111/jels.12413. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/jels.12413.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=aajyHYjjsk.

Dasha Metropolitansky and Jonathan Larson. VeriTrail: Closed-domain hallucination detection with traceability. In *The Fourteenth International Conference on Learning Representations*, pp. 39796–39838, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/42ec51d1031d25eca238ccc67559fe84-Paper-Conference.pdf.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12076–12100. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.741. URL https://doi.org/10.18653/v1/2023.emnlp-main.741.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. COVID-QA: A question answering dataset for COVID-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, jul 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.nlpcovid19-acl.18/.

Zakhar Mrykhin and Valentin Malykh. Prompt embedding probes (PEP): Hallucination detection in LLMs from hidden states. arXiv preprint arXiv:2608.08024, 2026. URL https://arxiv.org/abs/2608.08024.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=2DtxPCL3T5.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. In *The Twelfth International Conference on Learning Representations*, pp. 40364–40393, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/b0d4b9d3dc41b36cad1d36ddd27858ab-Paper-Conference.pdf.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10862–10878, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.585. URL https://aclanthology.org/2024.acl-long.585/.

Oscar Obeso, Andy Arditi, Javier Ferrando, Joshua Freeman, Cameron Holmes, and Neel Nanda. Real-time detection of hallucinated entities in long-form generation. arXiv preprint arXiv:2509.03531, 2025. URL https://arxiv.org/abs/2509.03531.

Yuya Ogasa and Yuki Arase. Hallucinated span detection with multi-view attention features. In *Proceedings of the 14th Joint Conference on Lexical and Computational Semantics*, pp. 381–394, Suzhou, China, nov 2025. Association for Computational Linguistics. ISBN 979-8-89176-340-1. doi: 10.18653/v1/2025.starsem-1.31. URL https://aclanthology.org/2025.starsem-1.31/.

OpenAI. GPT-5.6 Sol. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *The Thirteenth International Conference on Learning Representations*, pp. 66880–66913, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a712d461e57201efe35d429a6f1731c1-Paper-Conference.pdf.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 1310–1318. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/pascanu13.html.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Fabian Ridder and Malte Schilling. The HalluRAG dataset: Detecting closed-domain hallucinations in RAG applications using an LLM's internal states. In *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM)*, 2025. URL https://openreview.net/forum?id=pO0s74aF3R.

Fabian Ridder, Laurin Lessel, and Malte Schilling. RAGognizer: Hallucination-aware fine-tuning via detection head integration. arXiv preprint arXiv:2604.15945, 2026. URL https://arxiv.org/abs/2604.15945.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, Alex Wang, and Patrick Gallinari. QuestEval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6594–6604, Online and Punta Cana, Dominican Republic, nov 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.529. URL https://aclanthology.org/2021.emnlp-main.529/.

Adi Simhi, Itay Itzhak, Fazl Barez, Gabriel Stanovsky, and Yonatan Belinkov. Trust me, I'm wrong: LLMs hallucinate with certainty despite knowing the answer. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14665–14688, Suzhou, China, nov 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.792. URL https://aclanthology.org/2025.findings-emnlp.792/.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. LLM-Check: Investigating detection of hallucinations in large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 34188–34216. Curran Associates, Inc., 2024. doi: 10.52202/079017-1077. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/3c1e1fdf305195cd620c118aaa9717ad-Paper-Conference.pdf.

Hsuan Su, Ting-Yao Hu, Hema Swetha Koppula, Kundan Krishna, Hadi Pouransari, Cheng-Yu Hsieh, Cem Koc, Joseph Yitan Cheng, Oncel Tuzel, and Raviteja Vemulapalli. Learning to reason for hallucination span detection. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ECAK3P92eg.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, Bangkok, Thailand, aug 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.702. URL https://aclanthology.org/2024.acl-long.702/.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. Unsupervised real-time hallucination detection based on the internal states of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14379–14391, Bangkok, Thailand, aug 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.854. URL https://aclanthology.org/2024.findings-acl.854/.

Zhongxiang Sun, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Weijie Yu, Yang Song, and Han Li. ReDeEP: Detecting hallucination in retrieval-augmented generation via mechanistic interpretability. In *The Thirteenth International Conference on Learning Representations*, pp. 50250–50279, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/7daf60e805e596c3bd1e843e72ea5560-Paper-Conference.pdf.

Liyan Tang, Philippe Laban, and Greg Durrett. MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8818–8847, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.499. URL https://aclanthology.org/2024.emnlp-main.499/.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.330. URL https://doi.org/10.18653/v1/2023.emnlp-main.330.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023. URL https://arxiv.org/abs/2307.09288.

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. arXiv preprint arXiv:2307.03987, 2023. URL https://arxiv.org/abs/2307.03987.

Alex Wang, Kyunghyun Cho, and Mike Lewis. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5008–5020, Online, jul 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.450. URL https://aclanthology.org/2020.acl-main.450/.

Guangzhi Xiong, Zhenghao He, Bohan Liu, Sanchit Sinha, and Aidong Zhang. Toward faithful retrieval-augmented generation with sparse autoencoders. In *The Fourteenth International Conference on Learning Representations*, pp. 112549–112577, 2026. URL https://proceedings.

iclr.cc/paper_files/paper/2026/file/b6f6bfbd260fbf2f5acb0a1d6439ca0e-Paper-Conference.pdf.

Zitao Xuan, Xiaofeng Mao, Da Chen, Xin Zhang, Yuhan Dong, and Jun Zhou. ShieldHead: Decoding-time safeguard for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18129–18143, Vienna, Austria, jul 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.932. URL https://aclanthology.org/2025.findings-acl.932/.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, pp. 29687–29707, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4904fad153f6434a7bcf04465d4be2cc-Paper-Conference.pdf.

Samuel Yeh, Sharon Li, and Tanwi Mallick. LUMINA: Detecting hallucinations in RAG system with context-knowledge signals. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=oJgNNBNEJM.

Collin Zhang, Tingwei Zhang, and Vitaly Shmatikov. Speculative probing: LLM monitoring at speculative-decoding cost. arXiv preprint arXiv:2608.28099, 2026. URL https://arxiv.org/abs/2608.28099.

Fujie Zhang, Peiqi Yu, Biao Yi, Baolei Zhang, Tong Li, and Zheli Liu. Prompt-guided internal states for hallucination detection of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21806–21818, Vienna, Austria, jul 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1058. URL https://aclanthology.org/2025.acl-long.1058/.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59670–59684. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhang24ay.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Xiaoling Zhou, Mingjie Zhang, Zhemg Lee, Wei Ye, and Shikun Zhang. HaDeMiF: Hallucination detection and mitigation in large language models. In *The Thirteenth International Conference on Learning Representations*, pp. 81014–81038, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c98987c5ec4f30920d7190dc699e3daf-Paper-Conference.pdf.

Ilana Zimmerman, Jadin Tredup, Ethan Selfridge, and Joseph Bradley. Two-tiered encoder-based hallucination detection for retrieval-augmented generation in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 8–22, Miami, Florida, US, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.2. URL https://aclanthology.org/2024.emnlp-industry.2/.

A Additional Related Work

This section extends Section 2.

Benchmarks for Grounded Generation. Hallucination benchmarks label model responses as supported or unsupported by a source. HaluEval collects hallucinated and correct answers for question answering, dialogue and summarization (Li et al., 2023a). RAGTruth annotates unsupported spans in the responses of RAG systems (Niu et al., 2024), and TRIVIA+ adds long contexts from several question-answering sources with sentence-level human labels (Chen et al., 2026). FaithBench collects summaries from modern LLMs that existing detectors disagree on (Bao et al., 2025), and HalluRAG labels closed-domain hallucinations in RAG responses for probing the internal states of the generator (Ridder & Schilling, 2025). The responses in these benchmarks come from the LLMs that were available when the benchmarks were built. We keep their prompts and documents but generate new responses with current LLMs and label them anew.

External Verification. Verifiers compare a response with its evidence using a separate model. Early metrics ask and answer questions about a summary (Wang et al., 2020; Scialom et al., 2021) or apply entailment models to sentences (Laban et al., 2022). Fact checkers split a response into atomic claims (Min et al., 2023) or score pairs of document and claim with a small model (Tang et al., 2024). Lettuce labels unsupported tokens with a long-context encoder (Kovács & Recski, 2025), and RAGHalu checks each sentence of a response with two encoder classifiers for production customer-service systems (Zimmerman et al., 2024). TMSF classifies text, entailment and language-model features with a recurrent network (Itkin, 2026), and MVA extracts attention features with a proxy LLM to detect unsupported spans (Ogasa & Arase, 2025). LLM judges assess responses directly (Zheng et al., 2023), and recent verifiers learn to reason about unsupported spans with reinforcement learning (Su et al., 2026) or trace claims through the intermediate outputs of a multi-step process (Metropolitansky & Larson, 2026). Each of these verifiers loads a second model and processes the evidence again for every check. ProbeTok runs on the generator itself and checks each sentence with one additional input position and one forward pass per sentence of the response.

Uncertainty and Consistency. Uncertainty methods estimate whether the generator is unsure of its output. Token-level entropy gives a signal without additional samples (Malinin & Gales, 2021), and several methods verify or regenerate low-confidence parts of a response during generation (Varshney et al., 2023; Jiang et al., 2023). Sampling methods compare several responses to the same prompt (Manakul et al., 2023), cluster them by meaning (Kuhn et al., 2023; Farquhar et al., 2024) or search them for contradictions (Mündler et al., 2024). INSIDE measures the spread of the hidden states of sampled responses (Chen et al., 2024), and semantic entropy probes predict semantic entropy from the hidden states of a single generation (Kossen et al., 2024). However, the token probabilities of RLHF-tuned LLMs are often overconfident (Tian et al., 2023), and many hallucinations are produced with high certainty (Simhi et al., 2025). In our experiments, Min-Prob is a weaker detector than ProbeTok in every condition, as Appendix D shows. ProbeTok is trained to predict whether a sentence is supported by the documents rather than to estimate uncertainty, and it needs no additional samples at test time, since a single forward pass suffices.

Internal States. Internal-state detectors classify states that the generator has already computed. Probes on hidden states predict whether a statement is true (Azaria & Mitchell, 2023; Marks & Tegmark, 2024), and truthful directions can be found without labels (Burns et al., 2023) or used to steer generation (Li et al., 2023b). Truthfulness information concentrates in the exact answer tokens, and probes trained on one task transfer poorly to tasks that need other skills (Orgad et al., 2025). Later work combines hidden states with output statistics (Zhou et al., 2025) or projects them onto a reasoning subspace (Hu et al., 2026). For grounded generation, Lookback Lens uses the ratio of attention to the context and to the generated tokens (Chuang et al., 2024), and ReDeEP and LUMINA contrast the use of the context with the use of parametric knowledge (Sun et al., 2025; Yeh et al., 2026). RAGLens selects features of a sparse autoencoder (Xiong et al., 2026), and TPA decomposes the next-token probabilities into contributions from different sources (Lu et al., 2026b). LLM-Check computes scores from the hidden states, attention maps and logits of the generator or of a separate open LLM (Sriramanan et al., 2024). Trained detectors often lose accuracy in new domains, which PRISM addresses with prompts generated from a handwritten seed and selected on in-domain

data (Zhang et al., 2025) and SpikeScore with induced multi-turn continuations (Deng et al., 2026). All of these methods use states that were computed for next-token prediction, sometimes under a prompt. PROBETOK instead learns an input that makes the frozen generator compute a hidden state for detection, and Section 4.4 compares it with a handwritten question inserted after every sentence.

Learned Input Embeddings and Probing. Prompt tuning (Lester et al., 2021), prefix tuning (Li & Liang, 2021) and P-tuning (Liu et al., 2022) adapt a frozen model by learning continuous prompts. These prompts stay in the context and steer every later token toward the output of the new task. The probe token of PROBETOK is used at a single position and then removed, so the response is generated as it would be without detection. Concurrent work learns prompt embeddings that are inserted into the input of a frozen model, by default at its beginning, and trains a probe on the hidden states after a complete answer (Mrykhin & Malykh, 2026). Since the embeddings precede the question, the whole sequence must be processed again, and complete answers are scored after generation. PROBETOK appends its probe token at the end of a sentence, reuses the cache of the ongoing generation and scores each sentence while the response is being written. Other work reuses the cache of a frozen LLM for a single check. IntroLM (Kasnavieh et al., 2026) appends a token at prefill time to predict the quality of the upcoming answer, and KV-PRM (Kuang et al., 2026) appends a verify token after a reasoning step. Both use LoRA weights on the new token. Our Last-H and Fixed probe controls follow work on probing classifiers (Alain & Bengio, 2017; Belinkov, 2022) and are in the spirit of control tasks for probes (Hewitt & Liang, 2019), as are the two handwritten-question controls.

Architectures and the Generation State. Current open LLMs mix layer types to reduce the memory needed for long contexts. Sliding-window attention lets each token attend only to recent tokens (Beltagy et al., 2020), so a layer keeps only their keys and values. Linear attention (Katharopoulos et al., 2020; Yang et al., 2025) and state-space layers (Gu & Dao, 2024) replace the key-value history with a recurrent state. A detector that inserts an additional position must restore each of these states. Section 3 and Appendix C describe the restoration for the four LLMs in our experiments.

B DATA AND ANNOTATION

Datasets and Responses. RAGTruth provides question answering, summarization and data-to-text tasks with retrieved or given source material (Niu et al., 2024). Its questions come from MS MARCO (Bajaj et al., 2016), its articles include CNN/Daily Mail news (Hermann et al., 2015), and its structured records describe Yelp businesses. TRIVIA+ extends the evaluation of RAG to several question-answering sources (Chen et al., 2026). Its open-domain questions come from TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019) and MS MARCO. It also contains biomedical questions from COVID-QA (Möller et al., 2020) and questions from DROP (Dua et al., 2019) that require discrete reasoning such as counting or arithmetic. We use the prompts and documents of both datasets and generate new responses with the four LLMs, so every response set is specific to one generator. Table A1 gives the statistics of every split and condition.

Splits. We keep the test records designated by each dataset and split the remaining responses into training and validation sets by task or source. We then exchange responses between the training and validation sets until the validation set contains at least 150 unsupported sentences, so the share of unsupported sentences in the validation set differs from that in the test set. Every method selects its threshold on this validation set, so the thresholds are comparable across methods but are tuned at a higher share of unsupported sentences than the test set has. The two MS MARCO subsets of TRIVIA+ contain about 123 pairs of records per generator with the same prompt. No such pair is split between the training and test sets, since the test set follows the records that the dataset designates. Between 12 and 20 pairs per generator fall across the training and validation sets, which can make the validation AUROC slightly optimistic but does not affect the test set, and 16 to 17 pairs lie within the test set, where we keep both records as the dataset does. Other repeated question texts are generic DROP questions about different passages, such as which team scored first, and are not the same question. No question of RAGTruth appears in more than one split.

Annotation. The annotator is GPT-5.6 SOL at medium reasoning effort. It receives the prompt, the documents and the complete response and returns the unsupported claims of the response together with the words that make them unsupported, or it abstains. The annotator thus sees the complete

response, whereas real-time detectors such as PROBETOK see only the response generated so far. Whether a sentence is supported depends on the sentence and the documents, so the later sentences do not change its label. We exclude the responses on which it abstains, and we label a sentence as unsupported if it overlaps an unsupported span. As in RAGTruth, support is judged against the documents and not against world knowledge, so true but unstated content counts as unsupported.

Human Review. A human reviewer labeled 71 sentences, 43 of which the annotator had labeled unsupported. The sentences were drawn at random within each label from the responses of the four generators. The reviewer first judged each sentence and agreed with the annotator on 57 of the 71 sentences. For each of the 14 disagreements, the reviewer decided again with the judgment of the annotator shown, which is how the evaluators of Chuang et al. (2024) checked the labels of their LLM annotator. In 12 of them the annotator was right, usually about an unsupported detail such as a changed number or an event that the documents do not mention, and in the remaining two the reviewer applied a stricter reading than the annotator, so the final agreement is 69 of 71 sentences, or 97.2%. The annotator found errors that a careful human reader had overlooked, which supports the use of a recent LLM to label the responses of new generators. The supplementary material contains the reviewed sentences with both rounds of the judgments of the reviewer and the labels of the annotator.

Table A1: Statistics of the response sets on RAGTruth, denoted by RT, and TRIVIA+, denoted by TP. For each condition, we report the numbers of responses and sentences in the training, validation and test sets, and the number and percentage of unsupported sentences in the test set, denoted by H.

Condition	Responses train / val / test	Sentences train / val / test	Test H	Test H (%)
RT / Qwen	2,237 / 248 / 445	14,352 / 1,585 / 2,832	397	14.0
RT / Gemma	2,241 / 250 / 443	13,217 / 1,590 / 2,533	109	4.3
RT / Ministral	2,256 / 251 / 450	16,010 / 1,699 / 3,088	644	20.9
RT / Granite	2,252 / 251 / 448	17,308 / 1,942 / 3,348	462	13.8
TP / Qwen	1,747 / 195 / 468	12,641 / 1,634 / 3,387	373	11.0
TP / Gemma	1,506 / 168 / 416	7,608 / 962 / 2,134	141	6.6
TP / Ministral	1,497 / 167 / 407	16,924 / 2,358 / 4,710	1,598	33.9
TP / Granite	1,499 / 167 / 412	24,641 / 3,075 / 6,744	370	5.5

C IMPLEMENTATION DETAILS

Restoring the Cache. Algorithm 1 gives the order of operations. Since a period may belong to a number or an abbreviation, the splitter decides that s_i has ended only after the next token y_{n_i+1} has been sampled, and PROBETOK scores s_i before f_θ processes this token, while the cache is still $\mathcal{C}_{n_i}$. The probe token is never emitted as part of the response and is not kept in the cache. After scoring, PROBETOK restores every part of the cache that the additional position has changed. For full-attention layers, it removes the appended key and value. For sliding-window layers, it also puts back the entry that the probe token has evicted. For recurrent layers, it restores the saved recurrent state and the buffers of the short convolutions. It also restores the position counters. Shortening the cache is not enough for layers that overwrite their state, so their state is saved before the check and restored afterwards. Qwen3.5 combines full attention with Gated DeltaNet layers, Gemma-4 combines full and sliding-window attention, and Ministral-3 and Granite-4.2 use full attention only. In a live test with greedy decoding on 20 prompts for each LLM, including prompts longer than the sliding window of Gemma-4, the responses generated with and without the check were identical token for token, and the next-token logits did not differ. The cache after every restoration was equal to the cache before the check, tensor by tensor, in all 675 checks.

Training. During training, we feed each recorded response to the frozen LLM sentence by sentence and apply the probe token after every sentence, as in Algorithm 1. We initialize the probe token as $\mathbf{h}_{\text{probe}} = \gamma\mathbf{z}$, where the entries of $\mathbf{z}$ are drawn from a standard normal distribution and γ, listed in Table A2, is the mean standard deviation of the token embeddings of the LLM. We optimize $\mathbf{z}$ with

Algorithm 1 Hallucination detection with PROBETOK during generation

Require: Frozen LLM f_θ, prompt $\mathbf{x}$, probe token $\mathbf{h}_{\text{probe}}$, linear head $(\mathbf{w}, b)$ and standardization statistics

1: $\mathcal{C} \leftarrow$ cache of f_θ after processing $\mathbf{x}$
2: **while** generation has not ended **do**
3: Generate tokens until the sampled token y_{n_i+1} shows that sentence s_i has ended
4: Save the cache $\mathcal{C}_{n_i}$, which does not yet include y_{n_i+1}
5: $\mathbf{u}_i \leftarrow f_\theta(\mathbf{h}_{\text{probe}}; \mathcal{C}_{n_i})$
6: Output $p_i \leftarrow \sigma(\mathbf{w}^\top \bar{\mathbf{u}}_i + b)$ as the score of s_i
7: Restore the cache to $\mathcal{C}_{n_i}$, then process y_{n_i+1} and continue generation
8: **end while**

the probe learning rate divided by γ, so that the learning rate is relative to the scale of the embeddings. For Gemma-4, the per-layer embedding of the probe token is a 42×256 tensor, which we initialize to the mean per-layer embedding over the vocabulary and train together with $\mathbf{z}$. The mean and standard deviation used to standardize $\mathbf{u}_i$ in Equation 4 are computed once, before training, from the hidden states of 256 training sentences, and are kept fixed afterwards. Standard deviations below one-tenth of their median are raised to this value. The head is trained with AdamW, a learning rate of 10^{-3} and a weight decay of 0.1 or 0.5, and the probe token with no weight decay. We accumulate gradients over 64 sentences, warm up for 50 steps, decay the learning rate with a cosine schedule (Loshchilov & Hutter, 2017), clip the gradient norm at one (Pascanu et al., 2013) and train for at most 12 epochs with a patience of four epochs. The probe learning rate is one of two values for each LLM, listed in Table A2 together with the selected configurations.

Controls. Last-H uses the same standardization, head and training procedure as PROBETOK but classifies the last-layer hidden state of the last token of each sentence. Its configuration is selected on the validation set in the same way, and it is trained with three seeds that share this configuration and each select their epoch on the validation set. Fixed probe uses the configuration of PROBETOK, feeds the probe token at its random initialization and keeps all input parameters fixed, including the per-layer embedding of Gemma-4, so that only the head is trained on the same data at seed 58.

Handwritten-Question Controls. The two handwritten-question controls, P(Yes) and Question-H, insert the question "Does the last sentence of your response contain any claim that is not supported by the information provided above? Answer with Yes or No." after each sentence. The question is wrapped in the chat template of the generator as a user turn followed by the start of an assistant turn, appended to the cached prefix and removed from the cache after the check, exactly as the probe token is. Its wording was fixed before any result was seen, and it occupies 29 to 40 input positions depending on the tokenizer. P(Yes) uses no training. It reads the next-token distribution at the last position of the question and scores the sentence by $P(\text{Yes})/(P(\text{Yes}) + P(\text{No}))$, where each probability sums the single-token variants of the word. Question-H trains a linear head on the last-layer hidden state at the same position with the standardization, loss and optimizer of PROBETOK, selects its weight decay and epoch on the validation set as Last-H does, which chooses the larger weight decay of 0.5 in every condition since the two values lie within one standard error, and is averaged over three seeds. Table A3 reports the sentence-level and response-level AUROC of both controls for each of the eight conditions, together with the paired intervals of PROBETOK against the linear-head control. Averaged over three seeds, PROBETOK exceeds Question-H in five of the eight conditions, with an average of 0.763 against 0.748. P(Yes) falls below 0.5 for two RAGTruth generators, so its Yes probability is anti-correlated with the labels there, which may be because the question is inserted as a user turn in the middle of an unfinished response.

Configuration Selection. For each condition, we evaluate four configurations, which combine a low or high learning rate for the probe token with a weight decay of 0.1 or 0.5 for the head. For each configuration, we select the epoch with the highest validation AUROC. We then keep the configurations whose validation AUROC is within one standard error of the best, choose the one with the larger weight decay and break any remaining tie by validation AUROC. The standard error is estimated with the DeLong method (DeLong et al., 1988). We select the configurations of PROBETOK, Last-H and Question-H with seed 58. The selected configuration is shared by all

training seeds, and each seed selects its own epoch on the validation set. Table A2 lists the selected configurations. Test results are used neither to select the configuration nor to select the epoch.

Table A2: Configuration selected on the validation set for each condition. Config gives the learning rate of the probe token, low or high, and the weight decay of the head, 0.1 or 0.5. The probe learning rate is relative to the scale of the embeddings, whose mean standard deviation γ is given as Embedding SD. The learning rate of the head is 10^{-3} in all conditions. Validation AUROC is multiplied by 100, and the epoch and the validation AUROC are those of seed 58.

Condition	Config	Epoch	Val. AUROC	Probe LR	Embedding SD	Training seeds
RT / Qwen	high-wd05	4	74.4	5e-04	0.0135	58 / 7 / 123
RT / Gemma	low-wd05	4	71.7	5e-04	0.0224	58 / 7 / 123
RT / Ministral	high-wd05	9	71.5	2e-04	0.0055	58 / 7 / 123
RT / Granite	high-wd05	7	73.4	2e-02	0.5130	58 / 7 / 123
TP / Qwen	low-wd01	12	84.2	2e-04	0.0135	58 / 7 / 123
TP / Gemma	high-wd05	3	80.1	1e-03	0.0224	58 / 7 / 123
TP / Ministral	high-wd05	11	89.2	2e-04	0.0055	58 / 7 / 123
TP / Granite	high-wd05	6	85.3	2e-02	0.5130	58 / 7 / 123

Table A3: Handwritten-question controls, at the sentence level in the upper panel and at the response level in the lower panel. AUROC is multiplied by 100. P(Yes) and Question-H read the same handwritten question, through the logits of Yes and No and through a trained linear head, respectively. PROBETOK, Last-H and Question-H are averaged over three seeds, Fixed probe uses seed 58, and P(Yes) involves no training. The last two columns give PROBETOK minus Question-H in percentage points, with 95% paired bootstrap intervals from 1,000 resamples of the test responses in which the seeds are averaged within each resample. Bold marks the best AUROC in each row. A positive difference means that PROBETOK has the higher AUROC in that condition and at that level of evaluation, sentence or response, and a negative one that Question-H has it.

Condition	Handwritten question		Fixed probe	Last-H	ProbeTok	ProbeTok − Question-H	
	P(Yes)	Question-H				Diff. (pp)	95% CI (pp)
Sentence level							
RT / Qwen	52.0	69.9	64.3	65.7	**73.1**	+3.2	[+1.2, +5.3]
RT / Gemma	46.1	**74.6**	64.0	69.7	72.1	−2.5	[−5.4, +0.6]
RT / Ministral	57.3	**71.7**	67.4	64.7	68.4	−3.3	[−5.3, −1.3]
RT / Granite	49.6	71.0	63.8	68.1	**74.6**	+3.7	[+1.4, +5.8]
TP / Qwen	70.2	77.1	74.8	78.6	**79.0**	+2.0	[−0.8, +5.0]
TP / Gemma	64.5	**81.1**	70.7	77.0	79.8	−1.3	[−5.2, +2.2]
TP / Ministral	63.6	82.5	81.2	83.7	**84.2**	+1.7	[−0.6, +4.4]
TP / Granite	61.7	70.8	78.4	78.7	**79.4**	+8.5	[+5.1, +12.4]
Response level							
RT / Qwen	52.0	76.1	75.4	73.5	**78.9**	+2.7	[−0.2, +5.8]
RT / Gemma	56.5	71.7	69.5	70.6	**71.9**	+0.2	[−3.8, +4.2]
RT / Ministral	56.3	**73.3**	71.9	66.7	69.1	−4.2	[−9.3, +0.3]
RT / Granite	59.0	82.1	76.5	81.3	**84.7**	+2.6	[−0.3, +5.5]
TP / Qwen	**79.3**	76.2	75.5	78.1	77.8	+1.6	[−1.8, +4.9]
TP / Gemma	75.4	**77.0**	68.7	74.4	76.9	0.0	[−4.7, +4.5]
TP / Ministral	81.6	87.6	86.6	**90.2**	89.2	+1.6	[−1.1, +4.2]
TP / Granite	66.3	65.6	69.4	**77.9**	77.5	+11.9	[+6.5, +17.6]

D Details of the Baselines

Configurations. Lettuce uses the released large ModernBERT detector (Kovács & Recski, 2025). Lettuce-P encodes the complete response together with its documents, and Lettuce-R encodes the documents with the response up to the end of each sentence, once for every sentence. Lettuce-R selects its sentence and response thresholds on its own validation predictions with the same procedure as Lettuce-P. Our MiniCheck baseline uses Bespoke-MiniCheck-7B (Bespoke Labs, 2024), a later improvement of the original MiniCheck models (Tang et al., 2024), to score pairs of document and sentence. TMSF classifies text, entailment and language-model features with a bidirectional GRU (Itkin, 2026), and its released pipeline has no incremental version. MVA extracts attention features with Llama-3-8B-Instruct and applies a sequence classifier that reads the complete response (Ogasa & Arase, 2025). ReDeEP uses the weights of the generator to extract hidden, attention and vocabulary features (Sun et al., 2025), and LUMINA compares a pass with the real context and a pass with unrelated random documents (Yeh et al., 2026). The released implementations of both methods compute these features in separate passes with caching disabled. We retrain TMSF and MVA with three seeds on our data and use the released models of the other baselines without fine-tuning them.

Scores and Thresholds. For methods that score tokens, the score of a sentence is the maximum over its tokens, and the score of a response is the maximum over its sentences. We align the responses, sentences and labels of all methods within each condition. Each baseline uses the threshold selected on the validation set by its own procedure, and every threshold is fixed before the test set is evaluated.

Properties in Table 1. A detector runs in real time if the evaluated implementation can score each sentence as soon as the sentence is generated. A detector needs no extra LM if it loads no pretrained language model other than the generator, although it may have trained components such as a classifier. The properties describe the evaluated implementations, and other implementations of the same methods may differ. Appendix I describes how we measure the extra time.

Min-Prob. FLARE retrieves more documents when a token of the upcoming sentence has a probability below a threshold (Jiang et al., 2023), so its trigger is the lowest token probability of the sentence, which we call Min-Prob. We compute the probabilities with teacher forcing on the recorded test responses with the same encoding as in training, and score each sentence by one minus the lowest probability of its tokens. Table A4 also reports three other scores computed from the same probabilities, namely one minus the mean token probability, the mean entropy and the negative mean log-probability. Min-Prob is the best of the four in seven of the eight conditions, and ProbeTok outperforms all four in every condition. Since the generator computes these probabilities during generation anyway, Min-Prob adds no time in Table 1.

Table A4: Sentence-level AUROC of scores computed from the token probabilities of the generator, multiplied by 100, and of ProbeTok averaged over three seeds. Lowest probability is Min-Prob, one minus the lowest probability of a token in the sentence, which FLARE uses to decide when to retrieve, and Table 1 reports its average. The other scores are one minus the mean token probability, the mean entropy and the negative mean log-probability of the tokens in the sentence.

Condition	Lowest probability	Mean probability	Mean entropy	Mean log-probability	ProbeTok
RT / Qwen	66.4	62.4	63.2	62.9	73.1
RT / Gemma	66.6	60.6	62.3	61.3	72.1
RT / Ministral	61.6	57.8	58.7	58.2	68.4
RT / Granite	57.4	55.4	56.8	55.3	74.6
TP / Qwen	72.1	70.4	72.2	70.9	79.0
TP / Gemma	65.9	64.6	65.6	64.9	79.8
TP / Ministral	66.4	55.4	58.5	56.6	84.2
TP / Granite	60.9	58.7	60.4	58.1	79.4

E PAIRED COMPARISONS

We compare PROBETOK with the strongest baseline of each condition, level and metric. For a condition, a level ℓ, which is either the sentence or the response level, and a metric $\mu \in \{\mathrm{AUROC}, \mathrm{F1}\}$, let $M^{\mu,\ell}_{m,k}(I)$ denote the result of method m with training seed k on a set I of test responses. Let K_m be the set of seeds of method m, which contains a single element for the baselines with released models. We compare the means over seeds,

$$\bar{M}^{\mu,\ell}_{m}(I) = \frac{1}{|K_m|} \sum_{k \in K_m} M^{\mu,\ell}_{m,k}(I), \qquad \Delta^{\mu,\ell}(I) = \bar{M}^{\mu,\ell}_{\text{PROBETOK}}(I) - \bar{M}^{\mu,\ell}_{m^*}(I), \tag{6}$$

where m^* is the baseline with the best result on the full test set for this condition, level and metric. Each of the 5,000 bootstrap samples draws test responses with replacement and keeps all sentences of a response together, and all methods are evaluated on the same samples. We recompute AUROC and F1 for every seed on each sample, average over seeds and report the 2.5th and 97.5th percentiles of $\Delta^{\mu,\ell}$ as a 95% interval (Efron, 1979). The intervals account for the sampling of test responses. Table A5 lists all 32 comparisons. For the controls, Table A6 compares PROBETOK with Last-H, with both averaged over three seeds inside each of 1,000 bootstrap samples of the test responses, and Table A7 compares both with Fixed probe at seed 58 with 1,000 samples. The bootstrap treats the two records of a TRIVIA+ pair with the same prompt as separate responses, which concerns at most 4% of the TRIVIA+ test responses. Table A8 reports the variation over seeds.

Table A5: Difference between PROBETOK and the strongest baseline of each condition, level and metric, in percentage points. The strongest baseline is the one with the best result on the full test set. The 95% intervals come from 5,000 paired bootstrap resamples of the test responses, which keep the sentences of each response together in every resample.

Dataset	Generator	Level	Metric	Best baseline	Difference (pp)	95% CI (pp)
RAGTruth	Qwen	Sentence	AUROC	Lettuce-P	+1.46	[−1.5, +4.4]
RAGTruth	Qwen	Sentence	F1	Lettuce-P	+1.29	[−2.7, +5.5]
RAGTruth	Qwen	Response	AUROC	MVA	+1.37	[−1.7, +4.4]
RAGTruth	Qwen	Response	F1	TMSF	−1.72	[−4.3, +0.8]
RAGTruth	Gemma	Sentence	AUROC	Lettuce-P	−0.22	[−4.9, +4.3]
RAGTruth	Gemma	Sentence	F1	Lettuce-R	+0.02	[−4.6, +4.5]
RAGTruth	Gemma	Response	AUROC	LUMINA	+0.48	[−5.1, +6.0]
RAGTruth	Gemma	Response	F1	LUMINA	−0.32	[−3.9, +3.2]
RAGTruth	Ministral	Sentence	AUROC	Lettuce-P	−7.50	[−10.0, −5.1]
RAGTruth	Ministral	Sentence	F1	Lettuce-P	−7.71	[−10.9, −4.5]
RAGTruth	Ministral	Response	AUROC	TMSF	−9.41	[−13.7, −5.0]
RAGTruth	Ministral	Response	F1	Lettuce-R	−2.39	[−4.3, −0.5]
RAGTruth	Granite	Sentence	AUROC	Lettuce-R	+2.27	[−0.3, +5.0]
RAGTruth	Granite	Sentence	F1	Lettuce-R	+0.98	[−2.3, +4.4]
RAGTruth	Granite	Response	AUROC	TMSF	+1.23	[−1.5, +4.0]
RAGTruth	Granite	Response	F1	TMSF	+0.34	[−1.6, +2.3]
TRIVIA+	Qwen	Sentence	AUROC	LUMINA	+4.63	[+1.4, +7.7]
TRIVIA+	Qwen	Sentence	F1	Lettuce-P	+5.79	[+0.7, +11.1]
TRIVIA+	Qwen	Response	AUROC	LUMINA	−2.09	[−6.5, +2.2]
TRIVIA+	Qwen	Response	F1	TMSF	−3.32	[−7.6, +1.0]
TRIVIA+	Gemma	Sentence	AUROC	Lettuce-P	+3.39	[−1.9, +9.3]
TRIVIA+	Gemma	Sentence	F1	Lettuce-P	+4.39	[−5.2, +15.0]
TRIVIA+	Gemma	Response	AUROC	MiniCheck	+3.07	[−3.8, +9.9]
TRIVIA+	Gemma	Response	F1	MiniCheck	+4.67	[−4.2, +13.6]
TRIVIA+	Ministral	Sentence	AUROC	Lettuce-P	+6.32	[+2.8, +10.7]
TRIVIA+	Ministral	Sentence	F1	Lettuce-P	+1.67	[−1.3, +4.6]
TRIVIA+	Ministral	Response	AUROC	LUMINA	+2.34	[−0.8, +5.6]
TRIVIA+	Ministral	Response	F1	MiniCheck	−0.65	[−4.4, +3.2]
TRIVIA+	Granite	Sentence	AUROC	TMSF	+10.94	[+7.5, +14.3]
TRIVIA+	Granite	Sentence	F1	TMSF	+13.75	[+8.1, +19.7]
TRIVIA+	Granite	Response	AUROC	MiniCheck	+2.06	[−3.8, +8.0]
TRIVIA+	Granite	Response	F1	ReDeEP	+1.46	[−4.3, +7.2]

Table A6: Difference between PROBETOK and Last-H in percentage points, both averaged over three seeds, with 95% paired bootstrap intervals from 1,000 resamples of the test responses in which the seeds are averaged within each resample before the difference is taken.

Condition	Sentence Δ (pp)	Sentence 95% CI (pp)	Response Δ (pp)	Response 95% CI (pp)
RT / Qwen	+7.4	[+4.9, +10.2]	+5.4	[+1.9, +8.9]
RT / Gemma	+2.5	[−0.3, +5.4]	+1.3	[−1.5, +4.6]
RT / Ministral	+3.7	[+2.2, +5.4]	+2.4	[−1.4, +6.3]
RT / Granite	+6.5	[+4.6, +8.6]	+3.4	[+0.8, +6.2]
TP / Qwen	+0.5	[−1.3, +2.3]	−0.3	[−3.0, +2.4]
TP / Gemma	+2.8	[−0.1, +5.4]	+2.6	[−0.9, +5.7]
TP / Ministral	+0.5	[−1.5, +2.3]	−1.0	[−3.3, +1.5]
TP / Granite	+0.6	[−1.2, +2.6]	−0.5	[−3.9, +2.8]

Table A7: Paired differences between PROBETOK, Fixed probe and Last-H at seed 58 in percentage points, with 95% bootstrap intervals from 1,000 resamples of the test responses.

Condition	Level	ProbeTok − Fixed probe (pp)	95% CI (pp)	Fixed probe − Last-H (pp)	95% CI (pp)
RT / Qwen	Sentence	+9.16	[+6.4, +11.8]	−2.79	[−5.7, −0.4]
RT / Qwen	Response	+4.91	[+0.9, +8.9]	+0.22	[−3.6, +4.1]
RT / Gemma	Sentence	+11.49	[+6.6, +16.4]	−6.06	[−11.8, −0.3]
RT / Gemma	Response	+2.78	[−2.2, +7.4]	−2.84	[−7.7, +2.3]
RT / Ministral	Sentence	+0.84	[−1.0, +2.8]	+1.59	[−0.5, +3.6]
RT / Ministral	Response	−1.73	[−6.4, +2.8]	+6.66	[+1.5, +12.0]
RT / Granite	Sentence	+11.20	[+8.6, +13.6]	−6.06	[−8.8, −3.5]
RT / Granite	Response	+10.17	[+6.2, +14.7]	−7.67	[−11.8, −3.7]
TP / Qwen	Sentence	+6.80	[+3.7, +9.7]	−4.34	[−7.0, −1.7]
TP / Qwen	Response	+5.05	[+1.3, +8.8]	−3.84	[−7.5, −0.3]
TP / Gemma	Sentence	+9.14	[+4.8, +13.9]	−7.32	[−12.8, −2.3]
TP / Gemma	Response	+9.54	[+3.6, +15.7]	−6.14	[−12.5, −0.2]
TP / Ministral	Sentence	+3.31	[+1.7, +4.9]	−2.59	[−4.5, −0.6]
TP / Ministral	Response	+2.36	[−0.7, +5.3]	−3.97	[−7.2, −0.8]
TP / Granite	Sentence	−0.35	[−2.5, +1.7]	−0.08	[−2.3, +2.4]
TP / Granite	Response	+6.43	[+1.8, +11.1]	−7.03	[−11.6, −2.7]

F ADDITIONAL RESULTS

This appendix first gives the variation over seeds and two further comparisons of PROBETOK with Last-H, and then the thresholded metrics and the calibration of all detectors. Table A8 reports the sentence-level AUROC of each training seed for PROBETOK, TMSF and MVA. The standard deviation of PROBETOK over seeds is at most 0.029 and at most 0.012 in six of the eight conditions. Figure A1 plots the sentence-level AUROC of PROBETOK, Last-H and Fixed probe for each condition at seed 58. Averaged over three seeds, PROBETOK leads Last-H in eight of the eight conditions, and Table A6 gives the paired intervals. Scores of different responses can differ in scale, so Table A9 also compares PROBETOK and Last-H within responses. Such a comparison shows whether a detector finds the unsupported sentences of a given response. On the 1,445 test responses that contain both supported and unsupported sentences, PROBETOK ranks the unsupported sentences above the supported ones more often than Last-H in all eight conditions. Only PROBETOK ranks an unsupported sentence first in 280 responses, and only Last-H does so in 169.

Thresholded Metrics and Calibration. Tables A10 and A11 report the sentence-level precision and recall at the thresholds selected on the validation set, which correspond to the F1 scores in Table 2. Table A12 reports the results at a threshold that maximizes the validation recall at a false-positive rate of at most 5%, and Table A13 reports the calibration of the raw scores. The class weighting in Equation 5 shifts the scores of PROBETOK upward, so Table A13 also reports them after a temperature and a bias are fitted on the validation set, which leaves every AUROC unchanged, and after the logarithm of the class weight is subtracted from the logit, which needs no fitted parameter. The two corrections bring the average ECE of PROBETOK from 23.3 to 3.4 and 5.1, below the 11.0 of Lettuce-P and the 13.4 of Lettuce-R without any calibration. No score is calibrated with test labels, and all thresholds come from the validation set of each condition.

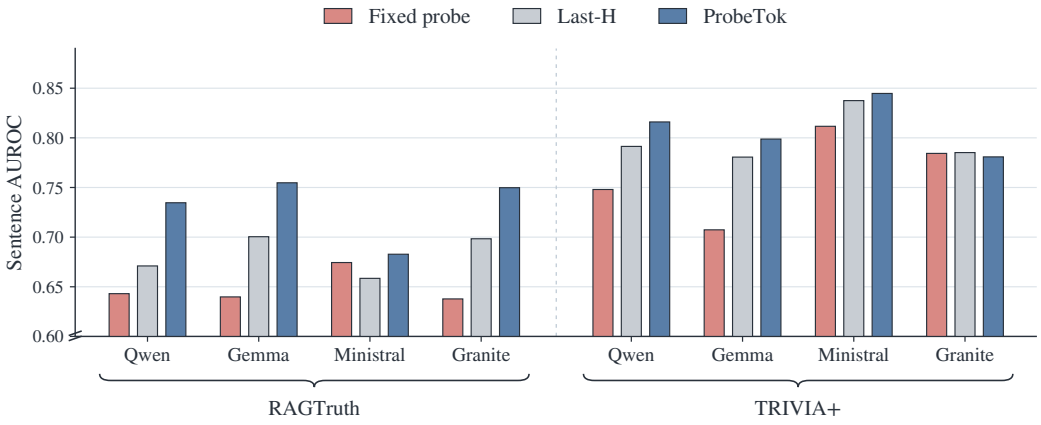


Figure A1: Sentence-level AUROC of Fixed probe, Last-H and PROBETOK. Fixed probe feeds the probe token at its random initialization and trains only the head, and Last-H classifies the last-layer hidden state at the last token of each sentence. The AUROC axis starts at 0.60. PROBETOK outperforms Fixed probe in seven of the eight conditions. Appendix E reports the paired differences, and Table 3 adds the two handwritten-question controls.

Table A8: Sentence-level AUROC of PROBETOK, TMSF and MVA for each training seed. All numbers are multiplied by 100, and SD is the standard deviation over the three seeds.

Dataset	Generator	Method	Seed 58	Seed 7	Seed 123	Mean $\pm$ SD
RAGTruth	Qwen	**ProbeTok**	73.5	72.9	73.0	73.1 $\pm$ 0.3
RAGTruth	Qwen	TMSF	63.2	65.4	63.5	64.0 $\pm$ 1.2
RAGTruth	Qwen	MVA	66.5	56.7	59.0	60.8 $\pm$ 5.1
RAGTruth	Gemma	**ProbeTok**	75.5	70.2	70.6	72.1 $\pm$ 2.9
RAGTruth	Gemma	TMSF	69.1	69.6	69.9	69.5 $\pm$ 0.4
RAGTruth	Gemma	MVA	65.6	48.3	72.1	62.0 $\pm$ 12.3
RAGTruth	Ministral	**ProbeTok**	68.3	69.0	68.0	68.4 $\pm$ 0.5
RAGTruth	Ministral	TMSF	64.8	63.1	66.3	64.8 $\pm$ 1.6
RAGTruth	Ministral	MVA	61.7	65.6	62.2	63.2 $\pm$ 2.1
RAGTruth	Granite	**ProbeTok**	75.0	74.3	74.5	74.6 $\pm$ 0.3
RAGTruth	Granite	TMSF	64.0	62.3	65.7	64.0 $\pm$ 1.7
RAGTruth	Granite	MVA	55.7	58.6	55.8	56.7 $\pm$ 1.6
TRIVIA+	Qwen	**ProbeTok**	81.6	77.7	77.8	79.0 $\pm$ 2.2
TRIVIA+	Qwen	TMSF	72.5	73.1	73.5	73.0 $\pm$ 0.5
TRIVIA+	Qwen	MVA	67.0	66.7	61.0	64.9 $\pm$ 3.4
TRIVIA+	Gemma	**ProbeTok**	79.9	79.9	79.7	79.8 $\pm$ 0.1
TRIVIA+	Gemma	TMSF	71.7	70.3	69.6	70.5 $\pm$ 1.1
TRIVIA+	Gemma	MVA	65.1	62.6	65.8	64.5 $\pm$ 1.7
TRIVIA+	Ministral	**ProbeTok**	84.5	84.4	83.7	84.2 $\pm$ 0.4
TRIVIA+	Ministral	TMSF	69.3	69.9	70.3	69.8 $\pm$ 0.5
TRIVIA+	Ministral	MVA	79.0	78.2	64.6	73.9 $\pm$ 8.1
TRIVIA+	Granite	**ProbeTok**	78.1	79.7	80.3	79.4 $\pm$ 1.2
TRIVIA+	Granite	TMSF	68.4	69.3	67.5	68.4 $\pm$ 0.9
TRIVIA+	Granite	MVA	65.7	65.5	67.6	66.3 $\pm$ 1.2

Table A9: Ranking of sentences within responses for PROBETOK and Last-H at seed 58. For each test response that contains both supported and unsupported sentences, we compute the AUROC of the unsupported against the supported sentences of that response and average it over responses, multiplied by 100. Top-1 is the percentage of these responses in which the highest-scored sentence is unsupported. Both detectors are evaluated on the same responses.

Condition	Mixed responses	ProbeTok AUROC	Last-H AUROC	ProbeTok top-1 (%)	Last-H top-1 (%)
RT / Qwen	236	67.6	62.1	43.6	34.3
RT / Gemma	86	68.6	65.4	34.9	31.4
RT / Ministral	314	67.0	66.3	47.1	41.7
RT / Granite	235	73.9	68.3	52.3	40.9
TP / Qwen	147	78.7	74.3	60.5	57.1
TP / Gemma	56	81.4	78.6	55.4	57.1
TP / Ministral	231	79.4	77.4	71.4	63.6
TP / Granite	140	75.0	70.0	49.3	35.0

Table A10: Sentence-level precision and recall on RAGTruth at the thresholds selected on the validation set. All numbers are percentages. PROBETOK, TMSF and MVA are averaged over three seeds. The best results for each generator are highlighted in bold. Precision and recall are computed on the test set at the same thresholds as the F1 scores in Table 2.

Generator	Method	Precision (%)	Recall (%)
Qwen	Lettuce-P	**29.7**	42.3
Qwen	Lettuce-R	27.3	47.4
Qwen	MiniCheck	23.8	57.4
Qwen	TMSF	18.8	64.1
Qwen	MVA	15.2	**94.0**
Qwen	ReDeEP	16.7	61.2
Qwen	LUMINA	17.8	77.1
Qwen	**ProbeTok**	26.9	56.8
Gemma	Lettuce-P	**13.9**	9.2
Gemma	Lettuce-R	10.5	28.4
Gemma	MiniCheck	10.0	18.3
Gemma	TMSF	8.8	28.4
Gemma	MVA	5.6	**83.8**
Gemma	ReDeEP	10.6	11.9
Gemma	LUMINA	7.1	41.3
Gemma	**ProbeTok**	8.9	56.0
Ministral	Lettuce-P	44.0	53.4
Ministral	Lettuce-R	**44.6**	50.0
Ministral	MiniCheck	33.3	61.3
Ministral	TMSF	26.7	71.9
Ministral	MVA	21.4	**98.7**
Ministral	ReDeEP	26.5	59.9
Ministral	LUMINA	24.8	83.1
Ministral	**ProbeTok**	28.3	71.8
Granite	Lettuce-P	25.9	56.3
Granite	Lettuce-R	**31.2**	41.8
Granite	MiniCheck	26.6	46.8
Granite	TMSF	19.3	57.9
Granite	MVA	14.2	**97.6**
Granite	ReDeEP	19.2	54.8
Granite	LUMINA	15.3	78.8
Granite	**ProbeTok**	27.8	60.0

Table A11: Sentence-level precision and recall on TRIVIA+ at the thresholds selected on the validation set. All numbers are percentages. PROBETOK, TMSF and MVA are averaged over three seeds. The best results for each generator are highlighted in bold. Precision and recall are computed on the test set at the same thresholds as the F1 scores in Table 2.

Generator	Method	Precision (%)	Recall (%)
Qwen	Lettuce-P	**34.6**	33.0
Qwen	Lettuce-R	19.0	52.0
Qwen	MiniCheck	18.3	51.5
Qwen	TMSF	31.4	35.7
Qwen	MVA	12.9	**90.3**
Qwen	ReDeEP	14.5	64.3
Qwen	LUMINA	25.0	46.9
Qwen	**ProbeTok**	32.2	52.0
Gemma	Lettuce-P	**29.4**	24.8
Gemma	Lettuce-R	13.2	**63.1**
Gemma	MiniCheck	13.6	14.2
Gemma	TMSF	14.1	37.6
Gemma	MVA	4.6	35.7
Gemma	ReDeEP	6.8	21.3
Gemma	LUMINA	11.2	46.8
Gemma	**ProbeTok**	21.9	58.2
Ministral	Lettuce-P	60.3	64.1
Ministral	Lettuce-R	66.8	57.3
Ministral	MiniCheck	49.4	74.7
Ministral	TMSF	54.0	57.5
Ministral	MVA	49.9	**78.5**
Ministral	ReDeEP	46.9	65.8
Ministral	LUMINA	47.0	75.0
Ministral	**ProbeTok**	**68.0**	60.4
Granite	Lettuce-P	22.9	13.2
Granite	Lettuce-R	**34.3**	9.7
Granite	MiniCheck	7.1	18.9
Granite	TMSF	22.9	15.0
Granite	MVA	20.2	39.5
Granite	ReDeEP	11.4	27.8
Granite	LUMINA	6.3	**48.1**
Granite	**ProbeTok**	26.4	37.7

Table A12: Results at a low false-positive rate, with PROBETOK at seed 58. The threshold τ maximizes the recall on the validation set subject to a false-positive rate (FPR) of at most 5%, so the false-positive rate on the test set can differ from 5%. The thresholds are on the scale of the raw scores, and the other numbers are percentages. RT denotes RAGTruth, and TP denotes TRIVIA+ throughout the table. Precision and recall are computed on the test set.

Dataset	Generator	Method	Val. τ	Test FPR (%)	Test recall (%)	Test precision (%)	Test F1 (%)
RT	Qwen	**ProbeTok**	0.850	4.9	16.9	36.0	23.0
RT	Qwen	Last-H	0.863	4.5	12.1	30.4	17.3
RT	Qwen	Lettuce-P	0.347	3.8	15.9	40.6	22.8
RT	Gemma	**ProbeTok**	0.836	4.1	15.6	14.7	15.1
RT	Gemma	Last-H	0.887	1.9	7.3	14.8	9.8
RT	Gemma	Lettuce-P	0.109	5.1	18.3	14.0	15.9
RT	Ministral	**ProbeTok**	0.702	4.8	16.9	48.2	25.1
RT	Ministral	Last-H	0.800	5.3	10.2	33.8	15.7
RT	Ministral	Lettuce-P	0.501	4.3	26.2	61.9	36.9
RT	Granite	**ProbeTok**	0.760	5.1	27.7	46.7	34.8
RT	Granite	Last-H	0.795	4.5	16.5	37.1	22.8
RT	Granite	Lettuce-P	0.611	4.8	19.5	39.5	26.1
TP	Qwen	**ProbeTok**	0.799	5.7	43.7	48.7	46.0
TP	Qwen	Last-H	0.841	4.7	31.4	45.0	37.0
TP	Qwen	Lettuce-P	0.896	3.1	20.1	44.4	27.7
TP	Gemma	**ProbeTok**	0.734	4.6	29.8	31.6	30.7
TP	Gemma	Last-H	0.800	4.5	29.8	32.1	30.9
TP	Gemma	Lettuce-P	0.788	4.3	25.5	29.8	27.5
TP	Ministral	**ProbeTok**	0.845	5.2	36.7	78.5	50.0
TP	Ministral	Last-H	0.839	6.4	34.1	73.4	46.6
TP	Ministral	Lettuce-P	0.935	6.0	34.9	74.9	47.6
TP	Granite	**ProbeTok**	0.951	3.3	24.6	30.1	27.1
TP	Granite	Last-H	0.752	3.6	24.3	28.3	26.2
TP	Granite	Lettuce-P	0.792	4.0	15.9	18.8	17.3

Table A13: Calibration of the raw scores at seed 58, measured by the expected calibration error (ECE) with 15 bins and the Brier score. All numbers are multiplied by 100, and lower is better. ReDeEP and LUMINA do not output probabilities and are therefore not evaluated on these two metrics. The two rows below PROBETOK report its scores after two corrections that leave every AUROC unchanged: Platt fits a temperature and a bias on the validation set, and prior-corrected subtracts the logarithm of the class weight in Equation 5 from the logit without any fitted parameter.

RAGTruth

	Qwen		**Gemma**		**Ministral**		**Granite**	
Method	ECE	Brier	ECE	Brier	ECE	Brier	ECE	Brier
TMSF	32.1	22.2	39.6	22.4	26.5	23.3	35.0	24.0
MVA	33.7	23.2	8.5	4.8	28.3	24.4	35.2	24.4
ReDeEP	–	–	–	–	–	–	–	–
LUMINA	–	–	–	–	–	–	–	–
Lettuce-P	10.3	12.6	3.7	4.5	11.4	15.2	9.6	12.4
Lettuce-R	9.7	12.6	3.8	4.6	11.0	15.2	9.6	12.3
MiniCheck	9.7	13.9	10.3	6.5	12.8	18.0	8.2	12.6
ProbeTok	42.1	30.5	31.5	19.8	15.9	18.9	18.6	16.5
Platt	1.0	11.0	4.3	4.2	3.7	15.4	1.3	10.3
prior-corrected	9.5	12.2	1.4	4.0	5.3	15.6	4.6	10.5

TRIVIA+

	Qwen		**Gemma**		**Ministral**		**Granite**	
Method	ECE	Brier	ECE	Brier	ECE	Brier	ECE	Brier
TMSF	33.8	21.1	35.8	20.4	15.8	22.0	28.5	16.5
MVA	31.6	19.5	5.7	6.5	20.3	23.3	42.3	23.0
ReDeEP	–	–	–	–	–	–	–	–
LUMINA	–	–	–	–	–	–	–	–
Lettuce-P	13.5	14.1	12.9	11.0	15.0	20.2	11.9	10.4
Lettuce-R	18.7	17.8	18.2	14.8	19.3	22.4	17.0	13.7
MiniCheck	40.7	33.1	32.8	21.5	28.2	28.1	37.5	23.9
ProbeTok	22.3	15.8	14.9	10.5	7.7	16.0	33.4	23.1
Platt	1.8	7.8	7.4	6.5	6.1	15.6	1.2	4.7
prior-corrected	2.9	7.9	3.2	5.7	7.0	15.8	6.8	6.6

G RESULTS BY TASK

Table A14 compares PROBETOK with the strongest baseline of each condition within every task, so that differences between tasks do not mix with differences in the scale of the scores. The clearest advantage of PROBETOK is on DROP, whose questions require counting, comparison or arithmetic over the passage. PROBETOK leads there by 0.068 to 0.160 in all four TRIVIA+ conditions, and every interval excludes zero. One possible reason is that an error in such a computation does not contradict any single sentence of the passage, which makes it hard to find for a detector that matches the response against the passage. PROBETOK also leads in most cells of MS MARCO, Natural Questions and TriviaQA. Summarization is its hardest task. The difference is negative for all four generators, from -0.026 to -0.086, although only the interval for RAGTruth–Ministral excludes zero. Summaries compress long articles, and an unsupported detail such as a date or an attribution appears among many faithful sentences and can be found only by checking it against the article. Of the 31 cells with at least one unsupported sentence, only two favor the baseline with an interval that excludes zero, and both belong to RAGTruth–Ministral, the condition analyzed in Appendix H.

Table A14: Sentence-level AUROC within each task for PROBETOK and the strongest baseline of each condition, multiplied by 100. Difference is PROBETOK minus the baseline in percentage points, with a 95% interval from 1,000 bootstrap resamples of the test responses. Bold marks the intervals that exclude zero among the differences with at least 20 unsupported sentences, and underlined differences rest on fewer than 20. The two MS MARCO subsets of TRIVIA+ share their questions and are merged into a single row of the table for each condition.

Condition	Baseline	Task	Unsupported sentences	ProbeTok	Baseline AUROC	Difference [95% CI]
RT / Qwen	Lettuce-P	Data-to-text	190	73.7	71.8	+1.9 [−3.2, +7.2]
		QA	35	68.1	69.7	−1.5 [−11.4, +7.6]
		Summarization	172	64.8	67.5	−2.6 [−7.3, +2.0]
RT / Gemma	Lettuce-P	Data-to-text	43	72.1	75.1	−3.0 [−9.5, +4.2]
		QA	5	66.6	58.8	+7.8 [−7.5, +29.5]
		Summarization	61	58.1	63.4	−5.3 [−13.9, +3.2]
RT / Ministral	Lettuce-P	Data-to-text	297	71.1	80.1	**−9.0 [−12.5, −5.5]**
		QA	126	70.1	75.1	−5.0 [−10.7, +0.5]
		Summarization	221	61.0	69.5	**−8.6 [−13.6, −3.7]**
RT / Granite	Lettuce-R	Data-to-text	244	79.8	77.9	+1.9 [−1.3, +5.0]
		QA	25	66.4	59.0	+7.5 [−2.8, +18.2]
		Summarization	193	62.4	66.0	−3.5 [−8.8, +1.6]
TP / Qwen	LUMINA	COVID	3	74.5	85.9	−11.4 [−31.7, +1.3]
		DROP	85	75.5	68.7	**+6.8 [+0.5, +12.7]**
		MS MARCO	141	75.7	70.2	+5.5 [−1.1, +11.8]
		NQ	96	80.3	79.9	+0.4 [−6.3, +6.9]
		TriviaQA	48	80.7	72.8	**+7.9 [+0.3, +15.1]**
TP / Gemma	Lettuce-P	COVID	0	–	–	–
		DROP	67	82.2	71.8	**+10.4 [+1.9, +19.8]**
		MS MARCO	36	77.3	81.9	−4.6 [−14.6, +5.7]
		NQ	18	73.8	75.1	−1.2 [−9.3, +16.3]
		TriviaQA	20	78.3	81.4	−3.2 [−14.8, +24.3]
TP / Ministral	Lettuce-P	COVID	52	85.2	86.8	−1.6 [−8.2, +16.8]
		DROP	177	78.9	65.9	**+13.0 [+6.8, +18.5]**
		MS MARCO	986	85.6	83.5	+2.2 [−0.3, +4.8]
		NQ	283	75.3	63.6	+11.7 [−2.5, +20.4]
		TriviaQA	100	83.6	79.3	+4.3 [−2.9, +11.4]
TP / Granite	TMSF	COVID	2	86.2	52.3	+33.9 [+21.2, +53.4]
		DROP	112	71.7	55.7	**+16.0 [+10.0, +22.1]**
		MS MARCO	157	81.5	70.3	**+11.2 [+5.9, +16.4]**
		NQ	59	77.7	71.0	+6.7 [−3.0, +18.3]
		TriviaQA	40	80.5	71.5	+9.1 [−6.2, +20.8]

H ANALYSIS OF RAGTRUTH–MINISTRAL

RAGTruth–Ministral is the only condition in which PROBETOK clearly trails the strongest detector, with an AUROC of 0.684 against 0.759 for Lettuce-P, whereas on the TRIVIA+ responses of the same generator PROBETOK reaches 0.842 and outperforms every detector. This appendix shows three things. The gap comes from this response set rather than from the Ministral model, it is not caused by an unusual mix of hallucination types, and it is concentrated in data-to-text and summarization, where Lettuce-P reaches its best results across the four RAGTruth response sets.

Attribution of the Gap to the Response Set. Two observations point to the responses rather than to the model. First, the gap appears for every training seed, with differences of -0.069, -0.076 and -0.080, whereas the differences for the other three RAGTruth response sets lie between -0.021 and

$+0.031$. Second, the responses of Ministral are also the hardest RAGTruth responses for PROBETOK running on the other LLMs. With Qwen as the detector, the AUROC is 0.668 on these responses against 0.692 to 0.744 on the other RAGTruth responses. With Granite, it is 0.656 against 0.697 to 0.746, and with Gemma, 0.599 against 0.614 to 0.721, as Figure 2 shows. Both detectors contribute to the gap: PROBETOK scores 0.049 below its mean on the other RAGTruth response sets, and Lettuce-P scores 0.039 above its own mean on those response sets and reaches its best RAGTruth result. Restricting Lettuce to the current prefix does not remove the gap, since Lettuce-R still reaches 0.750 on the same responses, against 0.684 for PROBETOK.

Types of Unsupported Sentences. The gap is not caused by an unusual share of some type of hallucination on which PROBETOK is weak. Table A15 splits the unsupported sentences along seven dimensions and compares each split with the RAGTruth responses of the other three generators. The dimensions are the task, the fraction of the sentence covered by unsupported spans, the error type recorded by the annotator, the format of the sentence, the entropy of the generator on the sentence, the lexical novelty of the sentence and its reliance on the documents. The error type is either a conflict with the evidence or an unsupported addition. Entropy is the mean token entropy of the sentence under its generator, and reliance is the decrease in the mean token log-probability of the sentence when the documents are replaced by filler text of the same length, as in Section 4.5. Both are computed with teacher forcing on the recorded responses, which are greedy outputs, since 99.9% of their tokens are the top prediction of the generator. Scoring 45 sentences per condition again with the same inputs reproduces the stored PROBETOK logits with a median absolute difference of 0.004. Each stratum of unsupported sentences is scored against all supported sentences of its condition, so the overall gap is the share-weighted sum of the gaps of the strata, and a difference in composition can be tested by reweighting the strata to the composition of another response set.

The strata in which PROBETOK is relatively weakest are largely the same across the four RAGTruth response sets, namely question answering, sentences whose wording does not appear in the prompt and sentences written with little reliance on the documents. For the other three generators, PROBETOK leads Lettuce-P in 27 of the 44 strata with at least 20 unsupported sentences, whereas for RAGTruth–Ministral it trails in all 16 such strata, including summarization and the sentences with high reliance, where it leads Lettuce-P on this measure for every other generator. Two compositions differ strongly, but neither explains the gap. Question answering makes up 20% of the unsupported sentences of RAGTruth–Ministral against 5% to 9% for the other generators, yet reweighting to the task mix of the other generators changes the gap only from -0.075 to -0.060. Likewise, 70% of its unsupported sentences are list items or marked-up sentences against at most 4% for the others, yet PROBETOK trails by the same margin on formatted and plain sentences. Moreover, 93% of the TRIVIA+ sentences of Ministral are also formatted, and PROBETOK leads on TRIVIA+–Ministral. On the remaining dimensions, reweighting leaves the gap between -0.073 and -0.085. The share of clear misses does stand out. We call an unsupported sentence a clear miss if PROBETOK ranks it below the median supported sentence while Lettuce-P ranks it above 80% of the supported sentences. Clear misses make up 10.2% of the unsupported sentences of RAGTruth–Ministral against 1.1% to 5.5% in the other seven conditions. The ranking within responses, which is unaffected by differences in scale between responses, shows the same gap. Table A16 gives a pair-weighted concordance of 68.4% for PROBETOK and 75.7% for Lettuce-P on RAGTruth–Ministral, against 83.1% and 80.7% on TRIVIA+–Ministral, where PROBETOK leads on the same measure.

Data-to-Text and Summarization. The gap is concentrated in two tasks. Within tasks, PROBETOK trails significantly only on data-to-text, by 0.090, and on summarization, by 0.086, as Table A14 shows. On both tasks, it stays close to its level for the other generators in Table A17, with 0.711 on data-to-text against 0.721 to 0.798 and 0.610 on summarization against 0.581 to 0.648. The gap arises because Lettuce-P reaches its best results on both tasks for the responses of Ministral, 0.801 against 0.718 to 0.775 on data-to-text and 0.695 against 0.634 to 0.675 on summarization. On data-to-text, part of the gap reflects the kind of error that Ministral makes. Of its unsupported data-to-text sentences, 27% misdescribe a business attribute such as the ambience or the parking, against 11% for Qwen and 18% for Granite. Such an error can be found only by checking the stated value against the structured record. Lettuce-P scores 0.88 on these sentences and PROBETOK 0.71, and Lettuce-P also leads on them for Qwen and Granite. Reweighting the errors of Ministral to the mix of Qwen and Granite lowers Lettuce-P to 0.779 and leaves PROBETOK unchanged, which removes about a quarter of the gap on data-to-text. On summarization, the mix of errors explains

Table A15: Decomposition of the gap between PROBETOK and Lettuce-P on RAGTruth–Ministral. The unsupported sentences are split into strata along each dimension, and each stratum is scored against all supported sentences of the condition, so that the overall gap of -7.5 points is the share-weighted sum of the gaps of the strata. Share is the percentage of the unsupported sentences in the stratum, and Gap is the AUROC of PROBETOK minus that of Lettuce-P in percentage points. Others gives the range over the RAGTruth responses of Qwen, Gemma and Granite, and its gap omits strata with fewer than 20 unsupported sentences. Reweighted gap applies the mean composition of these three response sets to the gaps of RAGTruth–Ministral. Entropy is split at the median token entropy of the supported sentences, and lexical novelty at the median share of content words that do not appear in the prompt. Reliance is split at the lower quartile of the decrease in log-probability when the documents are replaced, so that the low stratum holds the quarter of sentences least tied to the documents. The error-type split excludes eight unsupported sentences without an annotated claim, and shares may not sum to 100 because of rounding.

Dimension	Stratum	Share (%)	Others share (%)	Gap (pp)	Others gap (pp)	Reweighted gap (pp)
Task	Data-to-text	46	39–53	−3.7	−15.0 to +1.0	−6.0
	QA	20	5–9	−17.6	−23.0 to −5.7	
	Summarization	34	42–56	−6.8	+7.0 to +10.4	
Unsupported fraction	below 0.3	43	20–44	−4.5	−3.9 to +0.8	−8.5
	0.3 to 0.7	29	19–53	−7.2	−12.9 to +2.7	
	above 0.7	29	27–57	−12.3	+4.5 to +6.0	
Error type	conflict	34	38–50	−7.2	−0.6 to +5.7	−7.5
	addition	66	50–62	−7.7	+0.3 to +2.2	
Sentence format	list or markup	70	0–4	−7.5	–	−7.6
	plain	30	96–100	−7.6	−0.2 to +2.5	
Generator entropy	low	41	31–40	−6.1	−2.8 to +1.7	−7.6
	high	59	60–69	−8.5	+0.9 to +3.6	
Lexical novelty	new wording	64	56–62	−8.7	−6.8 to 0.0	−7.3
	source wording	36	38–44	−5.4	+3.7 to +10.7	
Reliance on documents	low	28	25–29	−10.4	−9.3 to −0.6	−7.4
	high	72	71–75	−6.3	+2.8 to +4.3	

none of the gap. Lettuce-P is stronger on all kinds of errors in the summaries of Ministral, most of all on dates and times, with 0.84 against 0.64. Dates and times are the errors on which PROBETOK trails most for the responses of Qwen and Granite as well.

Summary. The gap is not caused by an unusual share of any type of hallucination. It is concentrated in data-to-text and summarization, where PROBETOK performs close to its usual level and Lettuce-P reaches its best results. On data-to-text, the errors of Ministral are often attribute descriptions that must be checked value by value, which suits a token-level verifier. In the comparison within tasks, summarization is the only task on which PROBETOK trails for every generator, and the summaries of Ministral widen its gap to Lettuce-P beyond that of the other generators.

Table A16: Pair-weighted ranking of sentences within responses for PROBETOK and Lettuce-P on RAGTruth–Ministral and TRIVIA+–Ministral. For the responses that contain both supported and unsupported sentences, we report the percentage of pairs of an unsupported and a supported sentence from the same response in which the unsupported sentence receives the higher score. Ties between an unsupported and a supported sentence count as one half.

Condition	Mixed responses	Pairs	ProbeTok	Lettuce-P
RT / Ministral	314	3,497	68.4	75.7
TP / Ministral	231	28,479	83.1	80.7

Table A17: Sentence-level AUROC within each task on RAGTruth–Ministral, multiplied by 100. PROBETOK is averaged over three seeds, and Lettuce-P uses its released model.

Task	Sentences	Unsupported	ProbeTok mean ± SD	Lettuce-P	Gap (pp)
Data-to-text	1,213	297	71.1 ± 1.0	80.1	−9.0
QA	843	126	70.1 ± 0.7	75.1	−5.0
Summarization	1,032	221	61.0 ± 1.0	69.5	−8.6

I MEASUREMENT OF THE DETECTION TIME

For each condition, we sample 10% of the test responses and replay their generation one token at a time with batch size one on an L40 GPU, excluding two warm-up responses and the time to load the models. All detectors are timed on the same responses. For PROBETOK, the extra time is the time of the probe calls relative to the total time of generation,

$$\rho = \frac{\sum_r t_{\mathrm{probe},r}}{\sum_r \left(t_{\mathrm{prefill},r} + t_{\mathrm{decode},r}\right)}, \tag{7}$$

where r runs over the sampled responses, $t_{\mathrm{probe},r}$ is the total time of the probe calls in response r, and $t_{\mathrm{prefill},r}$ and $t_{\mathrm{decode},r}$ are the times to process the prompt and to generate the response. For the baselines, the numerator is the time of the detector on the same responses. The time of a probe call includes saving the cache before the check and restoring it afterwards, and the two handwritten-question controls are timed in the same way. All times are measured after the kernels have been warmed up, since the Gated DeltaNet layers of Qwen3.5 compile a kernel for each new sequence length within a process. For every prompt we run the prefill three times in one process and take the median of the last two runs, and for LUMINA and ReDeEP on Qwen3.5 we use their second pass over the response. A serving system pays this compilation once at startup. The extra time in Table 1 is the unweighted mean of ρ over the eight conditions, and Table A18 gives the value for each condition. PROBETOK adds between 3.0% and 5.1% across the eight conditions. Lettuce-R adds 7.1% because it encodes the documents and the response prefix again for every sentence, MiniCheck adds 18.7% because it runs a 7B model on every document–sentence pair, ReDeEP and LUMINA add 20.8% and 17.6% because they compute the internal states of the generator again, and Lettuce-P adds only 1.1% but can check the response only after generation has ended. Last-H and Min-Prob add no forward computation, since both read what the generator computes for the next token, so their extra time is 0.0%. Question-H and P(Yes) add 5.1% and 5.3%, at least as much as PROBETOK in every condition. Each check of PROBETOK adds one input position and each check of Question-H adds 29 to 40, so under batched serving, where the cost of a forward pass grows with the number of positions, the gap between the two widens. A single check takes 25 to 37 ms for PROBETOK, 38 to 80 ms for Lettuce-R and 55 to 675 ms for MiniCheck. The timed components differ between detectors. Lettuce-R sums the runs on successive prefixes, MiniCheck reuses the encoding of each document and includes a fallback for inputs that exceed its length limit, the times of TMSF and MVA exclude their final classifiers, and LUMINA runs two passes of the generator. The measurements therefore compare the main computation of each detector rather than the end-to-end latency of a complete serving system with batching and scheduling.

Table A18: Detection time as a percentage of the generation time for each condition, measured on an L40 GPU on the same 10% sample of the test responses after kernel warm-up. The generation time includes processing the prompt and generating the response, and the time to load the models is excluded. Question-H and P(Yes) are the two handwritten-question controls of Section 4.4. Appendix I describes which components are timed for each detector.

Method	**RAGTruth**				**TRIVIA+**			
	Qwen	Gemma	Ministral	Granite	Qwen	Gemma	Ministral	Granite
ProbeTok	5.0	4.2	3.5	3.3	4.7	5.1	4.6	3.0
Question-H	5.1	4.7	5.2	4.6	4.9	5.7	6.0	4.3
P(Yes)	5.5	4.9	5.4	4.7	5.3	6.0	6.3	4.4
Lettuce-P	1.2	1.0	0.9	0.9	1.8	1.3	0.8	0.9
Lettuce-R	7.1	5.2	7.1	5.7	10.3	5.5	7.8	7.9
MiniCheck	9.8	7.8	8.6	7.5	86.5	12.2	10.1	7.5
TMSF	2.5	1.6	2.0	1.9	1.8	1.8	2.1	1.5
MVA	6.2	5.3	6.1	4.8	12.9	7.0	4.5	5.0
LUMINA	16.2	16.8	16.8	14.4	23.9	25.2	14.9	12.5
ReDeEP	12.8	8.0	37.9	22.8	15.8	9.1	30.9	29.5

J Input Replacement

Table A19 gives the sentence-level AUROC of ProbeTok and Last-H before and after each replacement, at seed 58. When we replace the documents with filler text of the same length, we keep the instructions, the question and the response. When we replace the instructions in the same way, we keep the question, the documents and the response. Both detectors are evaluated with the same inputs in the same run, and each replacement is a separate run, so the AUROC on the full input is averaged over the two runs of a condition. A negative decrease means that the AUROC increased after the replacement. The AUROC of ProbeTok that remains after the documents are replaced, between 0.657 and 0.748, shows that some unsupported sentences can also be recognized from the question and the response alone. Figure A2 plots the decreases for both detectors and both replacements in each of the eight conditions at seed 58.

Counterfactual Documents. Replacing the documents with filler text shows that ProbeTok reads them, but not that it reads the evidence for the sentence being checked. We therefore edit a single fact. For each condition, we take 40 supported test sentences that state a fact of documents of at most 4,000 tokens, and GPT-5.6 SOL changes only that fact in the documents, the relevant edit, and separately an unrelated fact elsewhere in the documents, the irrelevant edit, each within a single span of at most 60 characters. The annotator of Appendix B then re-annotates the response under the edited documents, and we keep a relevant edit when the target sentence becomes unsupported, 256 of the 320 edits, and an irrelevant edit when it stays supported, 304 of the 320. The question and the response are fixed, and the score of the target sentence is read under the original and the edited documents with the seed 58 checkpoints. Table A20 reports the results. After a relevant edit, the logit of ProbeTok rises for 89.1% of the sentences, by 0.775 on average, whereas after an irrelevant edit it rises for 53.6% of them, close to chance, by 0.016 on average. Over the 242 pairs in which both edits are kept, the change under the relevant edit exceeds the change under the irrelevant edit by 0.785 on average, with a 95% interval of [0.645, 0.937], and the interval excludes zero in every condition. The rise does not come from earlier sentences that the edit may also have made unsupported: the logit rises for 89.3% of the 197 relevant edits after which no earlier sentence changed its label, and for 90.0% of the 70 target sentences that open the response. Last-H responds to the relevant edits as well, with a rise for 81.2% of the sentences and a paired difference of 0.736, and the two detectors do not differ significantly once their logits are standardized by their spread on the test set, with a larger effect for ProbeTok on RAGTruth and for Last-H on TRIVIA+.

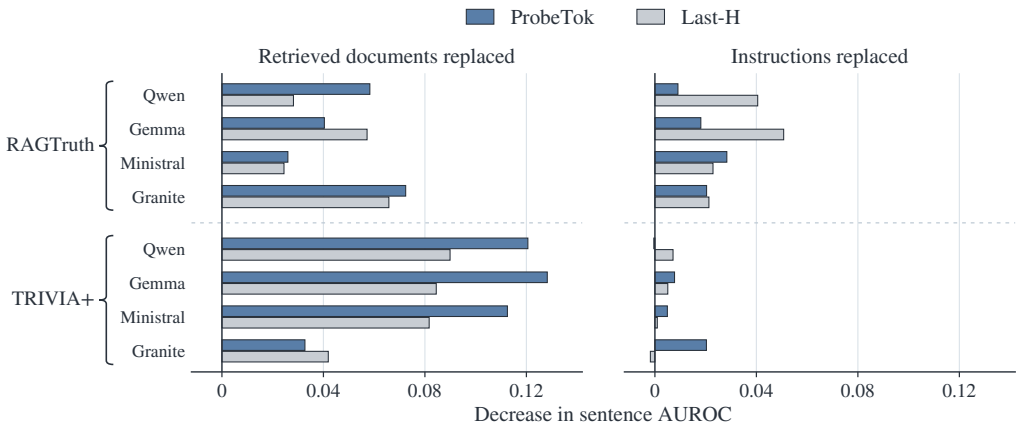


Figure A2: Decrease in sentence-level AUROC when parts of the prompt are replaced with filler text of the same length. Left: the retrieved documents are replaced. Right: the instructions are replaced, and the documents are kept. The question and the response are unchanged in both cases. Replacing the documents lowers the AUROC of PROBETOK in every condition and, in six of the eight, more than that of Last-H. Table A19 lists the AUROC before and after each replacement.

Table A19: Sentence-level AUROC before and after replacing the retrieved documents or the instructions with filler text of the same length, at seed 58. Decrease is the AUROC with the full input minus the AUROC after the replacement, so a negative value means that the AUROC increased.

Dataset	Generator	Detector	Replaced	Full AUROC	Replaced AUROC	Decrease
RAGTruth	Qwen	**ProbeTok**	Documents	0.735	0.676	+0.059
RAGTruth	Qwen	**ProbeTok**	Instructions	0.735	0.726	+0.009
RAGTruth	Qwen	Last-H	Documents	0.671	0.643	+0.028
RAGTruth	Qwen	Last-H	Instructions	0.671	0.631	+0.040
RAGTruth	Gemma	**ProbeTok**	Documents	0.755	0.714	+0.041
RAGTruth	Gemma	**ProbeTok**	Instructions	0.755	0.737	+0.018
RAGTruth	Gemma	Last-H	Documents	0.700	0.643	+0.057
RAGTruth	Gemma	Last-H	Instructions	0.700	0.650	+0.050
RAGTruth	Ministral	**ProbeTok**	Documents	0.683	0.657	+0.026
RAGTruth	Ministral	**ProbeTok**	Instructions	0.683	0.654	+0.029
RAGTruth	Ministral	Last-H	Documents	0.658	0.634	+0.024
RAGTruth	Ministral	Last-H	Instructions	0.658	0.636	+0.022
RAGTruth	Granite	**ProbeTok**	Documents	0.750	0.677	+0.073
RAGTruth	Granite	**ProbeTok**	Instructions	0.750	0.729	+0.021
RAGTruth	Granite	Last-H	Documents	0.698	0.633	+0.065
RAGTruth	Granite	Last-H	Instructions	0.698	0.677	+0.021
TRIVIA+	Qwen	**ProbeTok**	Documents	0.816	0.695	+0.121
TRIVIA+	Qwen	**ProbeTok**	Instructions	0.816	0.816	0.000
TRIVIA+	Qwen	Last-H	Documents	0.791	0.701	+0.090
TRIVIA+	Qwen	Last-H	Instructions	0.791	0.784	+0.007
TRIVIA+	Gemma	**ProbeTok**	Documents	0.799	0.671	+0.128
TRIVIA+	Gemma	**ProbeTok**	Instructions	0.799	0.791	+0.008
TRIVIA+	Gemma	Last-H	Documents	0.781	0.696	+0.085
TRIVIA+	Gemma	Last-H	Instructions	0.781	0.775	+0.006
TRIVIA+	Ministral	**ProbeTok**	Documents	0.845	0.732	+0.113
TRIVIA+	Ministral	**ProbeTok**	Instructions	0.845	0.839	+0.006
TRIVIA+	Ministral	Last-H	Documents	0.837	0.756	+0.081
TRIVIA+	Ministral	Last-H	Instructions	0.837	0.837	0.000
TRIVIA+	Granite	**ProbeTok**	Documents	0.781	0.748	+0.033
TRIVIA+	Granite	**ProbeTok**	Instructions	0.781	0.760	+0.021
TRIVIA+	Granite	Last-H	Documents	0.785	0.743	+0.042
TRIVIA+	Granite	Last-H	Instructions	0.785	0.787	−0.002

Table A20: Counterfactual documents at seed 58. For supported test sentences that state a fact of the documents, GPT-5.6 SOL changes only that fact, the relevant edit, or an unrelated fact, the irrelevant edit, and re-annotates the response. Relevant edits are kept when the sentence becomes unsupported and irrelevant edits when it stays supported. Up is the percentage of kept edits after which the logit of the sentence rises, and d is the change of the logit under the relevant edit minus the change under the irrelevant edit, averaged over the pairs in which both edits are kept, with a 95% bootstrap interval from 10,000 resamples of the pairs. Bold marks the intervals that exclude zero. PROBETOK and Last-H are read from the same forward pass of the generator.

Condition	Pairs	ProbeTok relevant up (%)	ProbeTok irrelevant up (%)	ProbeTok d [95% CI]	Last-H relevant up (%)	Last-H irrelevant up (%)	Last-H d [95% CI]
RT / Qwen	35	94.6	57.9	**+0.15 [+0.10, +0.21]**	67.6	50.0	**+0.20 [+0.07, +0.35]**
RT / Gemma	32	72.7	47.4	**+0.18 [+0.09, +0.30]**	87.9	57.9	+0.10 [−0.01, +0.20]
RT / Ministral	32	74.3	59.5	**+0.05 [+0.01, +0.09]**	57.1	56.8	+0.00 [−0.09, +0.08]
RT / Granite	34	97.3	43.2	**+1.82 [+1.24, +2.48]**	83.8	37.8	**+0.42 [+0.26, +0.59]**
TP / Qwen	30	90.6	47.4	**+1.05 [+0.73, +1.38]**	93.8	31.6	**+1.71 [+1.18, +2.29]**
TP / Gemma	29	93.3	59.0	**+1.10 [+0.74, +1.50]**	96.7	51.3	**+1.90 [+1.29, +2.54]**
TP / Ministral	22	90.9	64.1	**+0.75 [+0.47, +1.05]**	86.4	48.7	**+0.96 [+0.60, +1.33]**
TP / Granite	28	100.0	50.0	**+1.26 [+0.83, +1.73]**	83.3	63.2	**+0.94 [+0.49, +1.47]**
RAGTruth	133	85.2	52.0	**+0.56 [+0.37, +0.78]**	73.9	50.7	**+0.18 [+0.11, +0.26]**
TRIVIA+	109	93.9	55.2	**+1.06 [+0.87, +1.25]**	90.4	48.7	**+1.41 [+1.14, +1.70]**
All	242	89.1	53.6	**+0.79 [+0.65, +0.94]**	81.2	49.7	**+0.74 [+0.59, +0.90]**

K Details of Black-Box Detection and Transfer

Black-Box Setting. In Figure 2, the detector is the LLM that runs ProbeTok, and the generator is the LLM that produced the response. The detector processes the prompt and the response of the generator and scores each sentence as in Algorithm 1. It uses the probe token and head that were trained on its own responses to the same dataset, and no parameters or caches are shared between LLMs. The check therefore costs more than in the white-box setting, because the detector LLM must process the prompt and the response in addition to the generator. Table A21 reports the same values as Figure 2 with their standard deviations over the three training seeds. The averages in the main text are unweighted means over the six response sets of the other generators.

Other Datasets. We also evaluate each trained detector on the other dataset without retraining. Table A22 splits the RAGTruth responses into question answering, data-to-text and summarization, and compares, on the same sentences, the transferred detector with the detector trained on RAGTruth. The detectors trained on RAGTruth use all three tasks, so the transfer from RAGTruth to TRIVIA+ is reported for the pooled tasks. Question answering has the highest transferred AUROC for every detector LLM except Granite. A detector trained on TRIVIA+ scores the question-answering responses of RAGTruth within 0.016 in AUROC of a detector trained on RAGTruth itself when Qwen is the detector, and within 0.006 when Ministral is the detector, although Granite loses 0.116. On summarization and data-to-text, which do not appear in TRIVIA+, all four detectors lose between 0.043 and 0.240, and detectors trained on RAGTruth lose between 0.074 and 0.173 on TRIVIA+, whose questions come from other sources. The responses of Gemma contain only five unsupported sentences in RAGTruth question answering, so its large gain there rests on few examples. No question of RAGTruth question answering appears in TRIVIA+, no question of TRIVIA+ appears in RAGTruth, and the two datasets share no passage, so the transfer results in Table A22 do not rest on shared questions or shared documents between the two datasets.

Table A21: Sentence-level AUROC of each detector LLM, in rows, on the responses of each generator, in columns, multiplied by 100, with the mean and standard deviation over three seeds. The diagonal corresponds to the black Matched line in Figure 2.

Dataset	Detector	Qwen	Gemma	Ministral	Granite
RAGTruth	Qwen	73.1 ± 0.3	74.4 ± 0.6	66.8 ± 0.5	69.2 ± 0.7
RAGTruth	Gemma	63.8 ± 2.8	72.1 ± 2.9	59.9 ± 1.6	61.4 ± 3.3
RAGTruth	Ministral	63.7 ± 1.4	63.3 ± 2.6	68.4 ± 0.5	60.2 ± 1.7
RAGTruth	Granite	69.7 ± 0.4	72.3 ± 0.2	65.6 ± 1.1	74.6 ± 0.3
TRIVIA+	Qwen	79.0 ± 2.2	78.6 ± 2.9	77.8 ± 2.3	75.7 ± 2.3
TRIVIA+	Gemma	71.4 ± 0.4	79.8 ± 0.1	77.7 ± 3.1	65.8 ± 1.1
TRIVIA+	Ministral	76.7 ± 1.1	73.9 ± 2.1	84.2 ± 0.4	74.9 ± 1.7
TRIVIA+	Granite	76.6 ± 1.1	76.4 ± 2.5	80.3 ± 0.6	79.4 ± 1.2

Table A22: Transfer across datasets. We report the sentence-level AUROC of a detector trained on the target dataset and of a detector trained on the other dataset, evaluated on the same target sentences, multiplied by 100, with the mean and standard deviation over three seeds. Δ is the difference in percentage points, and N/H gives the numbers of target sentences and of unsupported ones. TP→RT denotes a detector trained on TRIVIA+ and evaluated on RAGTruth, shared marks the question-answering task that both datasets contain, and pooled means all three RAGTruth tasks together, since the detectors trained on RAGTruth use all three.

Detector	Direction / Task	N / H	In-domain AUROC	Transfer AUROC	Δ (pp)
Qwen	TP→RT / QA (shared)	637 / 35	68.1 ± 2.7	66.6 ± 6.5	-1.6 ± 4.4
Qwen	TP→RT / Data-to-text	1,306 / 190	73.7 ± 0.2	58.0 ± 4.7	-15.6 ± 4.5
Qwen	TP→RT / Summarization	889 / 172	64.8 ± 0.3	53.9 ± 3.7	-10.9 ± 3.9
Qwen	RT→TP / pooled	3,387 / 373	79.0 ± 2.2	70.1 ± 1.2	-9.0 ± 2.9
Gemma	TP→RT / QA (shared)	477 / 5	66.6 ± 5.1	82.9 ± 6.4	$+16.3 \pm 4.8$
Gemma	TP→RT / Data-to-text	1,241 / 43	72.1 ± 4.2	50.3 ± 6.5	-21.7 ± 3.3
Gemma	TP→RT / Summarization	815 / 61	58.1 ± 4.4	53.8 ± 5.0	-4.3 ± 2.4
Gemma	RT→TP / pooled	2,134 / 141	79.8 ± 0.1	62.5 ± 6.7	-17.3 ± 6.6
Ministral	TP→RT / QA (shared)	843 / 126	70.1 ± 0.7	69.4 ± 2.0	-0.6 ± 1.9
Ministral	TP→RT / Data-to-text	1,213 / 297	71.1 ± 1.0	56.5 ± 1.4	-14.6 ± 1.0
Ministral	TP→RT / Summarization	1,032 / 221	61.0 ± 1.0	55.4 ± 2.9	-5.6 ± 3.9
Ministral	RT→TP / pooled	4,710 / 1,598	84.2 ± 0.4	71.5 ± 5.1	-12.7 ± 4.8
Granite	TP→RT / QA (shared)	436 / 25	66.4 ± 1.7	54.8 ± 1.3	-11.6 ± 2.9
Granite	TP→RT / Data-to-text	1,836 / 244	79.8 ± 0.7	55.8 ± 3.5	-24.0 ± 2.8
Granite	TP→RT / Summarization	1,076 / 193	62.4 ± 0.4	52.9 ± 1.7	-9.5 ± 1.5
Granite	RT→TP / pooled	6,744 / 370	79.4 ± 1.2	71.9 ± 0.8	-7.4 ± 1.2

L CONFIDENT HALLUCINATIONS

This appendix reports how each detector performs on the unsupported sentences that the generator wrote with high confidence. We call an unsupported sentence a confident hallucination if the generator assigned a mean probability of at least 0.9 to the words that make it unsupported. These words are the error spans that the annotator marked for the unsupported claims in the sentence. When the annotator marked no error span, we use the span of the claim in the response. We compute the probabilities with teacher forcing on the recorded responses, using the same encoding as in training, and we check the token range of every sentence against the stored one. On average, 27% of the unsupported sentences are confident hallucinations, and this share ranges from 8% to 50% across conditions. For token probabilities, we use in each condition the best of the four scores in Table A4, which favors this baseline over any single one of the four scores in every condition.

Table A23 reports the share of confident and other unsupported sentences that each detector flags when its threshold flags 5%, 10% or 20% of the supported sentences. At every rate, token probabilities flag a smaller share of the confident hallucinations than of the supported sentences, so on these sentences they do no better than a random score. The other detectors, except LUMINA, flag confident hallucinations at rates close to their rates on the other unsupported sentences.

We also evaluate the detectors by AUROC, using the confident hallucinations as positives and all supported sentences of the same condition as negatives. Token probabilities reach 0.49 on average, between 0.41 and 0.55 across conditions, compared with 0.65 on all unsupported sentences. PROBETOK reaches 0.74, between 0.63 and 0.80, compared with 0.76 on all unsupported sentences. Last-H at seed 58 reaches 0.74, and the strongest baseline of each condition reaches 0.70. The results are similar for other thresholds. With a threshold of 0.8, confident hallucinations make up 53% of the unsupported sentences, and the AUROC is 0.56 for token probabilities, 0.75 for PROBETOK, 0.74 for Last-H and 0.72 for the strongest baseline. With a threshold of 0.95, they make up 15%, and the corresponding values are 0.46, 0.73, 0.72 and 0.69.

Table A23: Percentage of unsupported sentences that each detector flags when its threshold flags 5%, 10% or 20% of the supported sentences of the same condition. Confident denotes the unsupported sentences whose unsupported words the generator wrote with a mean probability of at least 0.9, and Other denotes the remaining unsupported sentences. Values are averaged over the eight conditions, PROBETOK, TMSF and MVA are also averaged over three seeds, and Last-H uses seed 58. Token probability uses the best of the four scores in Table A4 for each condition. A detector that ignores its input flags about the same percentage of unsupported and supported sentences.

Method	Real time	Confident 5%	Other 5%	Confident 10%	Other 10%	Confident 20%	Other 20%
Token probability	Yes	4.4	14.0	7.9	25.2	13.9	40.9
MiniCheck	Yes	13.7	17.1	25.2	26.1	40.5	43.8
Lettuce-R	Yes	21.0	25.5	34.0	36.7	48.8	50.6
Last-H	Yes	22.0	22.1	36.5	33.8	54.9	52.8
ProbeTok	Yes	22.5	27.5	35.3	40.0	54.0	57.0
Lettuce-P	No	21.2	25.8	33.9	37.4	48.7	52.1
TMSF	No	17.0	18.8	28.1	28.3	41.0	43.9
MVA	No	16.0	13.4	23.7	21.6	35.9	35.7
LUMINA	No	7.7	13.4	12.6	23.3	23.7	39.1
ReDeEP	No	11.0	10.2	20.5	19.2	33.1	34.1

M LIMITATIONS AND FUTURE WORK

We note five limitations of this work. First, PROBETOK is trained with sentence labels for each generator and dataset. We obtain these labels from a recent LLM, and a human review found them reliable. Section 4.6 also shows that a detector trained on the responses of one generator can check the responses of the other three generators, so a new generator does not always require new labels, although we have tested this only among these four generators. Second, token-level verifiers such as Lettuce remain competitive on summarization and on data-to-text responses with many attribute values, as Appendices G and H show. When a response can wait for a complete check, such a verifier can complement PROBETOK. Third, our experiments use English datasets and open LLMs with up to 9B parameters, the black-box setting is evaluated only among these four generators, and the detection time is measured by replaying generation with batch size one rather than in a serving system, so training and evaluating PROBETOK on larger generators, on other languages and under batched serving is left for future work. Fourth, the raw scores of PROBETOK are not calibrated, since the class weighting in Equation 5 shifts them upward. A temperature and a bias fitted on the validation set bring the average ECE from 23.3 to 3.4, below the 11.0 of Lettuce-P and the 13.4 of Lettuce-R without any calibration, and leave every AUROC unchanged, as Table A13 reports. Fifth, we have not varied the number of probe tokens or the layer that the head reads, and both are natural extensions.

The main purpose of real-time detection is to correct an unsupported sentence before it reaches the user. In future work, we plan to connect PROBETOK to such a correction step, which retrieves more documents or regenerates the sentence when PROBETOK flags it, as FLARE does with token probabilities. This closed loop would turn the detector into a method for mitigating hallucinations. We plan to measure how much it reduces the hallucinations in the final response and how much delay the check and the correction add to the response in a serving system.