SCOUT: TOKEN-EFFICIENT DETECTION OF MODEL SUBSTITUTION IN LLM APIS

Anonymous authors
Paper under double-blind review

ABSTRACT

When people use a large language model (LLM) through an API, they cannot see which model actually answers their requests. A provider can quietly swap the advertised model for a cheaper sibling or a quantized variant. Current defenses are either unreliable or too expensive to run often. We frame the task as model difference probing: detecting whether an endpoint's per-question success probabilities differ from those of a trusted reference. We propose SCOUT, a sequential auditor designed to use few tokens to tackle this. SCOUT models the unknown substitute as a Bayesian mixture of public benchmark models. At each step, it asks the question expected to best separate the served model from the reference per token spent. It then uses the answer to update both its posterior over the substitute and an anytime-valid test statistic that keeps the false-alarm rate below a chosen level α. On the Open LLM Leaderboard and on real commercial APIs, SCOUT is more accurate than all prior methods while using over 90% fewer tokens.

1 INTRODUCTION

Large language models (LLMs) are increasingly accessed through APIs rather than deployed locally. This convenience limits transparency: users cannot inspect the model behind an endpoint. A provider may therefore silently replace an advertised model with a cheaper or less capable alternative, or selectively degrade its service. Verifying that an API continues to provide the advertised capability is thus important. Existing response-based fingerprinting (Pasquini et al., 2025), adversarial-prompt fingerprinting (Gubri et al., 2024), and implanted instructional fingerprints (Xu et al., 2024) are useful in their intended settings, but they do not necessarily distinguish closely related models or certify their capabilities. Our comparisons in Section 3.1 show that the evaluated identity checks can be unreliable for within-family substitutions, including quantized variants: response patterns or provenance signals may persist even when task performance changes.

Benchmark evaluation offers a direct alternative: compare an endpoint with a trusted reference model on graded tasks. It can reveal changes caused by a smaller replacement, quantization, or other modifications, without requiring a fingerprint tailored to every possible substitute. However, comprehensive benchmarking requires substantial latency, queries, and token usage. Efficient evaluation methods (Perlitz et al., 2024; Maia Polo et al., 2024b; Kipnis et al., 2025) reduce this cost by estimating full-benchmark scores from fewer examples, but our objective is different: quickly deciding whether one endpoint differs from one reference. A question that estimates overall ability well may not distinguish two similar models, so even compact evaluations can spend resources on uninformative questions (Section 3.2). This setting raises two challenges: selecting questions that are most likely to expose a difference under a limited budget, and making a reliable sequential decision despite ordinary generation variability. Because repeated evaluations of the same model can differ, a single disagreement is insufficient evidence; the procedure must control false alarms under adaptive question selection and early stopping.

We address these challenges with SCOUT, which treats each graded response as evidence about the model served by an endpoint. SCOUT maintains a Bayesian posterior over possible served models, represented as a mixture of publicly evaluated benchmark models; selects the question with the greatest expected evidence per token; and accumulates answers in an anytime-valid e-value, a sequential evidence statistic that remains valid under adaptive selection and early stopping. It returns DIFFERENT, SAME, or INCONCLUSIVE. Unlike identity checks, SCOUT can detect same-family or

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.

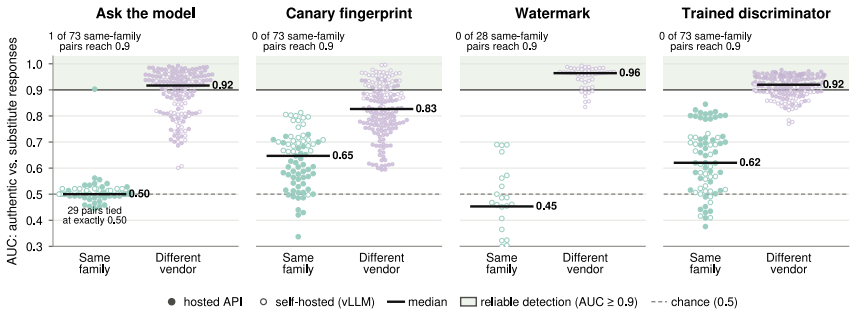


Figure 1: **Identity checks fail on same-family substitutions.** Each point is one (reference, substitute) pair. All four checks detect different-vendor substitutes (median AUC 0.83–0.96) but fall to near chance within a family (median 0.45–0.65), below the 0.9 threshold for reliable detection.

quantized substitutes when they change task performance. Unlike efficient benchmarking, it selects questions to distinguish this endpoint from this reference and stops once the evidence is sufficient, reducing token consumption by up to 90% in our experiments (Section 5.2). A rule that stops at the first unexpected answer falsely flags every genuine endpoint in our experiments; SCOUT instead weights each answer by the reference model's reliability on that question and controls false alarms under any selection rule and stopping time. Its posterior determines what to ask, whereas its e-value determines what to conclude; an inaccurate posterior can slow an audit but cannot create a false alarm.

Our contributions are:

- We show that existing identity checks can fall to near-chance detection for realistic same-family substitutions, while efficient benchmarking methods require many queries and tokens to reach one audit decision.
- We propose a Bayesian sequential testing framework that models each answer as a draw from a model-specific, per-question success probability. It maintains a posterior over the served model and accumulates evidence in an e-value whose false-alarm rate is controlled under any question selection rule and stopping time.
- We introduce a token-aware selection algorithm that chooses the question with the highest expected evidence for a difference per token. Its token price is selected automatically using Dinkelbach's method rather than hand-tuned. Compared with a random question order under the same test, it reduces token consumption by up to 90%.

2 Related Work

Model Identity Detection. Prior work verifies model identity or provenance using behavioral fingerprints (Gubri et al., 2024; Pasquini et al., 2025; Bruckner, 2026; Fang et al., 2026), watermarks and embedded or representation-based fingerprints (Kirchenbauer et al., 2023; Xu et al., 2024; Zhang et al., 2025; Hu et al., 2026), or statistical tests of output consistency with a trusted reference (Gao et al., 2025; Zhu et al., 2026; Dima et al., 2025; Cai et al., 2025b). These methods establish identity, provenance, or distributional consistency, but do not directly guarantee equivalent task performance. We instead detect reproducible, question-level differences in success probabilities under a low token budget, including for same-family and quantized substitutions.

Efficient Benchmark Evaluation. Efficient evaluation estimates benchmark performance from informative subsets, item response theory, coreset selection (Perlitz et al., 2024; Maia Polo et al., 2024a; Kipnis et al., 2025; Yao et al., 2026; Maia Polo et al., 2024c), or adaptive question selection and sampling (Hofmann et al., 2025; Li et al., 2026; Truong et al., 2025; Wu et al., 2026). Pairwise methods select questions that separate two specific models (Feng et al., 2025). These approaches aim to estimate or rank overall performance, whereas we aim to detect a capability difference between a reference and a candidate using as few tokens as possible. A question that is informative for overall performance may not distinguish two models when both answer it equally well; we therefore combine adaptive, token-aware probing with sequential testing.

Table 1: Cost for existing methods to decide whether an endpoint serves the reference model, averaged over 3,000 held-out model pairs from the Open LLM Leaderboard v2. *Original size*: number of questions each method uses in its original paper. *Queries* and *Tokens*: average number of questions asked and input-plus-output tokens consumed before a decision.

Method	Original size	Queries	Tokens
Efficient benchmark evaluation			
tinyBenchmarks (Maia Polo et al., 2024a)	200	23.4	23.1k
ATLAS (Li et al., 2026)	500	16.9	25.2k
metabench (Kipnis et al., 2025)	858	27.1	28.6k
FAQ (Wu et al., 2026)	1,500	42.6	56.0k
Output-distribution tests			
Model Equality Testing (Gao et al., 2025)	25	18.4	24.3k
Rank-Based Uniformity Test (Zhu et al., 2026)	100	104.2	141.5k

3 MOTIVATION

This section explains why existing methods cannot detect model substitution both reliably and cheaply. Throughout, an **endpoint** is an API access point offered by a provider. The **reference model** is the model the provider claims to serve. The **served model** is the model actually serving the endpoint and under testing now, which may be the reference model or an unknown substitute.

3.1 IDENTITY CHECKS MISS SAME-FAMILY SUBSTITUTIONS

A user paying for a large model through an API cannot see which model serves each request. A provider could substitute a smaller sibling or a quantized variant while still charging for the advertised model. A natural defense is to check the endpoint's identity. Behavioral fingerprints can tell models from different vendors apart (Gubri et al., 2024; Pasquini et al., 2025), but models in the same family may give the same identifying responses even when their task performance differs.

Identity checks. We evaluate four checks (details in Appendix A). **Ask the model** asks the endpoint for its model name or version and compares the answer with the advertised model (Cai et al., 2025a). **Canary fingerprint** sends fixed diagnostic prompts and compares the answers with those registered from the reference model (Gubri et al., 2024; Pasquini et al., 2025). **Watermark** tests outputs for a keyed token pattern inserted during generation, which requires the watermark key and tokenizer (Kirchenbauer et al., 2023). **Trained discriminator** trains a classifier on labelled responses from the reference model and a proposed substitute, then classifies new responses (Sun et al., 2025).

Results. We test each check on 73 endpoints from 14 model families under two kinds of substitution: within a family (e.g., Doubao-Seed-2.0 $\rightarrow$ Doubao-Seed-2.0-lite) and across vendors (e.g., Doubao-Seed-2.0 $\rightarrow$ DeepSeek-V4). In Figure 1, each point is one (reference, substitute) pair, scored by the AUC for separating reference responses from substitute responses (0.5 is chance). Across vendors, all four checks work well, with median AUC between 0.83 and 0.96. Within a family, they fall to near chance, with median AUC between 0.45 and 0.65. Only one same-family pair, under any check, reaches the 0.9 AUC we treat as reliable detection. **Identity checks therefore cannot reliably detect the most plausible substitution: a cheaper model from the same family.**

3.2 CAPABILITY TESTING WORKS, BUT EFFICIENT BENCHMARKS ARE TOO EXPENSIVE

Section 3.1 shows that identity checks cannot reliably separate a model from a same-family substitute. Such a substitute can keep the reference model's identifying responses while answering some questions correctly with a different probability. Benchmark questions with verifiable answers expose this difference directly. An audit does not need to estimate everything either model can do; it needs reliable evidence of a reproducible difference. The converse does not hold: agreement on a finite set of questions cannot prove that two models are identical.

Efficient benchmarks answer a different question. Efficient benchmark methods reduce evaluation cost while preserving a model's score, ability estimate, or ranking (Perlitz et al., 2024; Maia Polo et al., 2024a; Kipnis et al., 2025; Li et al., 2026; Wu et al., 2026). However, a question that helps estimate overall performance gives little evidence about a specific pair if both models answer it alike. Detecting a substitution needs questions that separate the pair, not a reconstruction of each model's full benchmark score.

Their cost is too high for frequent audits. Table 1 measures this cost on 3,000 held-out model pairs from the Open LLM Leaderboard v2 (Fourrier et al., 2024). To decide whether the candidate model is the reference model, efficient benchmark methods need 17–43 queries (23K–56K tokens), and output-distribution tests need 18–104 queries (24K–142K tokens). The cost recurs. A provider can swap the model at any time, for example to a cheaper sibling during peak load, and swap it back later. Because the auditor cannot know when a swap happens, the endpoint must be checked often, so each check must use few tokens and few sequential queries. **Efficient benchmarks are built to estimate overall performance, not to separate a specific pair, so each audit decision costs tens of queries and tens of thousands of tokens, too much for frequent checks.**

4 METHOD

We audit an API endpoint that claims to serve a reference model. The key difficulty is that LLM generation is stochastic: the reference can fail a question it usually solves, so treating the first unexpected answer as a substitution would create many false alarms. We therefore compare the endpoint's and reference's *performance distributions* while asking as few questions as possible. SCOUT alternates two steps (Figure 2). *Query selection* (Section 4.2) chooses the question expected to provide the most evidence per token. *Belief update and sequential testing* (Section 4.3) uses the answer to update the predicted endpoint behavior, test whether it matches the reference, and decide whether to stop. The loop ends when it reaches a decision or exhausts the token budget.

4.1 PROBLEM SETUP

Assumptions. We assume a benchmark with a pool $\mathcal{Q}$ of N questions with verifiable answers and a set $\mathcal{M}$ of benchmark models. For every model $m \in \mathcal{M}$ and question $q \in \mathcal{Q}$, we have a profile $p_m(q) \in (0, 1)$: the probability that m answers q correctly. Keeping these values strictly between 0 and 1 ensures that one unexpected answer never rules out a model completely. For the reference model R, we record k_q correct answers from n_q runs on each question and the average token cost $\hat{c}_q$ (input plus output). These profiles come from public leaderboards and one offline profiling run of the reference, and are reused across checks. During a check, we query only the endpoint, using a fixed prompt, decoding configuration, and answer parser.

Notation. Let $p_R(q)$ denote the reference's true success probability on q. We estimate it by $\hat{p}_q = (k_q + 1)/(n_q + 2)$, which remains in $(0, 1)$ and is near $1/2$ when the reference is unstable or poorly measured. We call the unknown model serving the endpoint the *substitute model* C, with success probabilities $p_C(q)$. At step $t = 0, 1, 2, \ldots$, we send q_t to the endpoint, receive output o_t, and record its correctness $y_t \in \{0, 1\}$:

$$y_t \sim \mathrm{Bern}\big(p_C(q_t)\big), \qquad \mathrm{Bern}(y; p) = p^y (1 - p)^{1-y}.$$

Hypothesis. The endpoint serves the reference when both models have the same question-level success probabilities. We test

$$H_0 : \; p_C(q) = p_R(q) \text{ for all } q \in \mathcal{Q} \qquad \text{against} \qquad H_1 : \; p_C(q) \neq p_R(q) \text{ for some } q \in \mathcal{Q}. \quad (1)$$

We test question-level differences rather than only the benchmark average, because a substitute may lose a few specific capabilities while preserving its overall score.

4.2 QUERY SELECTION THROUGH OPTIMIZATION

Objective. Our goal is to reach a decision with minimal total token use. At step t, we score each unasked question by

$$U_t(q) = G_t(q) - \lambda_t \, \hat{c}_q, \quad (2)$$

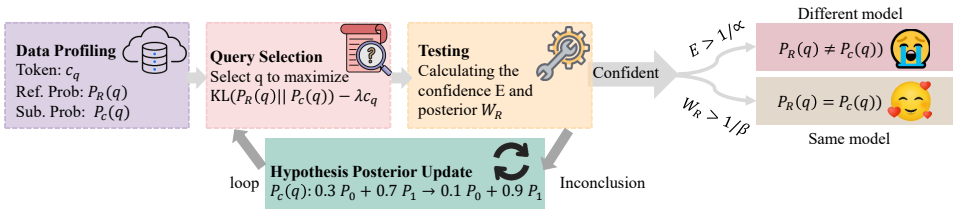


Figure 2: SCOUT overview. Before auditing, we profile the question-level success probabilities of the reference and benchmark candidate models. At each step, their weighted mixture predicts substitute behavior and selects the question with the greatest expected evidence per token. After grading the endpoint response, SCOUT updates the mixture posterior and sequential evidence, then either returns a decision or continues.

where $G_t(q)$ is the expected evidence for separating the endpoint from the reference, $\hat{c}_q$ is the token cost, and λ_t is the current token price in units of evidence.

Choosing λ_t. The tokens required for a decision are roughly the needed evidence divided by the evidence gained per token (Wald, 1945). We therefore set the token price to the best currently available evidence-per-token rate, $\lambda_t = \max_q G_t(q)/\hat{c}_q$. With this choice, the maximum of Eq. (2) is zero and is attained by exactly the question with the largest ratio $G_t(q)/\hat{c}_q$. This is the standard ratio-optimization device used in Dinkelbach's method (Dinkelbach, 1967) (Appendix A). Thus, no hand-tuned λ is needed: a question is selected only when its expected evidence justifies its token cost.

Substitute model estimation. To estimate $G_t(q)$, we need to predict how the endpoint would answer if it were not the reference. Because the substitute is unknown, we use a weighted mixture of benchmark models:

$$P_{C,t}(q) = \sum_{m \in \mathcal{M}} w_{m,t}\, p_m(q), \qquad w_{m,t} \geq 0, \qquad \sum_{m \in \mathcal{M}} w_{m,t} = 1. \tag{3}$$

If the endpoint serves an unlisted model, the weights move toward benchmark models whose answers best match it, so the mixture can still approximate its performance. An imperfect approximation can slow a check, but cannot invalidate the false-alarm guarantee (Proposition 1). The reference is excluded from this mixture because it represents substitute behavior; the probability that the endpoint is the reference is tracked separately (Section 4.3). The initial weights $w_{\cdot,0}$ form a prior. A uniform prior averages all benchmark models, whereas an informed prior gives more weight to plausible substitutes, such as siblings or quantized variants, and none to implausible alternatives, such as more expensive models.

Separating power. We define

$$G_t(q) = \mathrm{KL}\big(P_{C,t}(q) \,\|\, \hat{p}_q\big), \qquad \mathrm{KL}(a \,\|\, b) = a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b}, \tag{4}$$

the KL divergence between the predicted substitute and reference answer distributions. If the substitute behaves as predicted, $G_t(q)$ is the expected increase in $\log E_t$ from asking q (Eq. (6); Appendix A). It is zero when the predictions agree and increases as they diverge. For the same probability gap, it is larger when the reference is stable ($\hat{p}_q$ near 0 or 1) than when it is noisy or poorly measured ($\hat{p}_q$ near $1/2$). Under a uniform prior, SCOUT therefore favors inexpensive questions that the reference nearly always solves while other models often fail, or the reverse. As answers arrive, the weights concentrate on a few candidates, and selection shifts from separating the reference from all benchmark models to separating it from the leading candidates.

4.3 BELIEF UPDATE AND SEQUENTIAL TESTING

After each endpoint response, SCOUT updates its substitute prediction, updates the evidence against the reference hypothesis, and decides whether to stop.

Updating the belief. We update the mixture weights by Bayes' rule according to how well each model predicted the observed correctness:

$$w_{m,t+1} = \frac{w_{m,t}\ \mathrm{Bern}\big(y_t;\, p_m(q_t)\big)}{\sum_{m' \in \mathcal{M}} w_{m',t}\ \mathrm{Bern}\big(y_t;\, p_{m'}(q_t)\big)}. \tag{5}$$

A correct answer increases the weights of models that usually solve q_t; an incorrect answer does the reverse. The larger the disagreement among models on q_t, the larger the update. Over time, $P_{C,t}$ moves closer to the substitute's actual performance distribution.

Posterior estimation with a small language model. Correctness provides only one bit of evidence per query, so Eq. (5) can distinguish similar models slowly. The full output o_t also contains formatting, phrasing, and reasoning-style cues that may reveal a model family. We therefore train a small language model g_ϕ on held-out outputs from the benchmark models to estimate the producing model. Given (q_t, o_t), it returns a score $g_\phi(m \mid q_t, o_t)$ for each $m \in \mathcal{M}$. We multiply the Bayesian weight in Eq. (5) by $g_\phi(m \mid q_t, o_t)^\gamma$, where $\gamma \in [0, 1]$ tempers the reader's confidence (Appendix B). Because this reader only redistributes weight among substitute candidates, it changes the alternative prediction but not the reference side of the test. It can therefore accelerate an audit but cannot create a false alarm.

Testing. We accumulate evidence that the endpoint differs from the reference in a running e-value E_t. Each check starts with $E_0 = 1$ and, after each answer, updates

$$e_t = \frac{\mathrm{Bern}\big(y_t;\, P_{C,t}(q_t)\big)}{\mathrm{Bern}\big(y_t;\, \hat{p}_{q_t}\big)}, \qquad E_{t+1} = E_t\, e_t. \tag{6}$$

The factor e_t compares how well the substitute prediction and reference profile explain the observed answer. An unexpected answer carries more weight when the reference is reliable on that question: for example, a wrong answer where the reference almost always succeeds can multiply E_t by up to $1/(1 - \hat{p}_{q_t})$, whereas a question with $\hat{p}_{q_t}$ near $1/2$ contributes at most about 2 (and symmetrically for correct answers). If the substitute prediction is accurate, $\log E_t$ grows by $G_t(q_t)$ per query in expectation, motivating the query-selection objective.

Proposition 1 (No false alarms beyond α)**.** *Suppose the reference profile is accurate, $\hat{p}_q = p_R(q)$ for every question, and let each question q_t and prediction $P_{C,t}$ depend only on the outputs before step t. Then, under H_0,*

$$\mathbb{P}\left(E_t \geq \frac{1}{\alpha} \text{ for some } t\right) \leq \alpha.$$

Under H_0, the endpoint answers q_t correctly with probability $\hat{p}_{q_t}$, so each factor e_t has conditional expectation one. Thus, E_t is a fair sequential bet: a genuine endpoint cannot increase it on average. Ville's inequality bounds the probability that it ever reaches $1/\alpha$ (Ville, 1939; Grünwald et al., 2024) (Appendix C). The result holds for any adaptive question rule, stopping time, prior, and reader g_ϕ, because these choices only shape the predictable alternative $P_{C,t}$. A poor prediction may slow an audit, but cannot make it falsely declare a difference. When $\hat{p}_q$ approximates rather than equals $p_R(q)$, the bound degrades smoothly with the estimation error (Appendix C).

Confidence and stopping. The check stops when either conclusion is sufficiently confident:

- **Different.** Define $\mathrm{Conf}_t = 1 - 1/\max_{s \leq t} E_s$. By Proposition 1, Conf_t reaches $1 - \alpha$ for a genuine endpoint with probability at most α. We stop when $E_t \geq 1/\alpha$ and report the benchmark model with the largest weight as the most likely substitute.
- **Same.** Let $w_{R,t}$ be the posterior probability that $C = R$, with prior π_R. Since E_t is the likelihood ratio of the substitute and reference predictions, the posterior odds satisfy

  $$\frac{w_{R,t}}{1 - w_{R,t}} = \frac{\pi_R}{(1 - \pi_R)\, E_t}$$

  (Appendix D). We stop when $w_{R,t} \geq 1 - \beta$.

Otherwise, the two steps repeat until the budget is exhausted, and the check returns INCONCLUSIVE. Each check restarts from $E_0 = 1$, so α controls the false-alarm probability of each individual check. Algorithm 1 summarizes the procedure.

Algorithm 1 One SCOUT check.

Require: costs $\hat{c}_q$, reference profile $\hat{p}_q$, and model profiles $p_m(q)$; prior weights $w_{\cdot,0}$ and π_R; levels α, β; token budget $C_{\max}$

$E_0 \leftarrow 1$
for $t = 0, 1, 2, \ldots$ **do**
  Compute $P_{C,t}(q) \leftarrow \sum_{m \in \mathcal{M}} w_{m,t} p_m(q)$ for each unasked q ▷ Eq. (3)
  $\lambda_t \leftarrow \max_q G_t(q)/\hat{c}_q$ and $q_t \leftarrow \arg\max_q \{G_t(q) - \lambda_t \hat{c}_q\}$ ▷ Eqs. (2) and (4)
  Send q_t to the endpoint; observe output o_t and correctness y_t
  $E_{t+1} \leftarrow E_t e_t$ and $w_{R,t+1} \leftarrow \pi_R / \big(\pi_R + (1 - \pi_R) E_{t+1}\big)$ ▷ Eq. (6)
  Update $w_{m,t+1}$ ▷ Eq. (5) and g_ϕ
  if $E_{t+1} \geq 1/\alpha$ **then**
    return DIFFERENT, Conf_{t+1}, most likely substitute
  end if
  if $w_{R,t+1} \geq 1 - \beta$ **then**
    return SAME, $w_{R,t+1}$
  end if
  if tokens spent $\geq C_{\max}$ **then**
    return INCONCLUSIVE, Conf_{t+1}, $w_{R,t+1}$
  end if
end for

5 EVALUATION

5.1 EXPERIMENTAL SETUP

Benchmark and models. We use question-level correctness records from the Open LLM Leaderboard v2 (Fourrier et al., 2024), covering IFEval, Big-Bench Hard, MATH Level 5, GPQA, MuSR, and MMLU-Pro. Of 4,500 model evaluations over 21,606 items, we retain the 4,479 with complete records, ensuring that missing scores are never treated as incorrect answers. We split these models into 4,031 historical and 448 held-out models. All methods use the items supported by every baseline, split at random 9:1 into 19,423 evaluation items and 2,158 items used only to train the output reader g_ϕ on historical models.

Audit trials. Each historical model serves in turn as the reference. For each reference, we run one genuine trial, where the endpoint serves the reference itself, and one substitution trial for every other historical or held-out model. The mixture in Eq. (3) and all learned components use only the remaining historical models. Thus, a historical substitute appears among the candidates, whereas a held-out substitute is unknown to the auditor and tests the unlisted substitute setting (Section 4.2). Appendix E details the model profiles.

Baselines. We compare SCOUT with uniform sampling and the selection policies of metabench (Kipnis et al., 2025), tinyBenchmarks (Maia Polo et al., 2024a), ATLAS (Li et al., 2026), and Factorized Active Querying (FAQ) (Wu et al., 2026), all fitted on the historical models. To isolate selection quality, these policies receive the same profiles and candidate models and use the same sequential test and stopping rules (Section 4.3); they are published selection policies adapted to auditing. We separately report binary-output adaptations of Model Equality Testing (Gao et al., 2025) and the Rank-Based Uniformity Test (Zhu et al., 2026), which retain their own reference-sampling requirements and multiple-testing corrections. The uniform baseline applies McNemar's test: it samples questions uniformly, queries both the reference and endpoint, and compares the counts of questions answered correctly by only one of them. It returns DIFFERENT when the difference is significant, SAME at the query cap, and otherwise continues.

Parameters. Unless stated otherwise, we use $\alpha = 0.05$, $\beta = 0.01$, and $\pi_R = 0.5$ in Algorithm 1. We sweep α and β over $\{0.1, 0.05, 0.01, 0.001\}$ in Section 5.4. We report a full stopping rule, which can return SAME, and a detection-only rule, which cannot. Token-budget experiments use nine budgets $C_{\max}$ from 1,000 to 256,000 estimated tokens, doubling at each step; every check is capped at 100 queries.

5.2 Overall Evaluations

We compare SCOUT with the baselines in Section 3. Each baseline is calibrated to a false-positive rate of at most 5%.

Method	FPR↓ (%)	Detect.↑ (%)	Queries ↓ (avg.)	Tokens ↓ (avg.)	Tokens ×SCOUT
SCOUT (ours)	**2.0** ±0.2	**97.2** ±0.7	**3.7** ±0.3	**1.4k** ±0.2k	**1**
RUT (Zhu et al., ICLR'26)	4.5 ±0.6	94.5 ±1.6	105.1 ±9.7	142.9k ±7.6k	105.1
MET (Gao et al., ICLR'25)	2.5 ±0.2	78.9 ±7.9	18.0 ±1.3	24k ±27.6k	177
ATLAS (Li et al., ICML'26)	3.9 ±0.8	81.2 ±5.8	16.4 ±2.1	24.6k ±1.2k	18.1
metabench (Kipnis et al., ICLR'25)	4.0 ±0.7	81.1 ±9.7	27.5 ±1.8	28.0k ±4.1k	20.6
Uniform (baseline)	4.2 ±0.4	72.7 ±4.6	32.8 ±4.4	44.4k ±4.9k	32.8
FAQ (Wu et al.)	3.9 ±0.7	79.4 ±8.8	41.7 ±6.1	55.1k ±4.8k	40.9
tinyBenchmarks (Maia Polo et al., ICML'24)	4.0 ±0.5	81.7 ±8.2	22.9 ±3.1	22.6k ±2.4k	16.6

Table 2: Accuracy and cost on the Open LLM Leaderboard v2. *FPR*: genuine endpoints declared DIFFERENT; *Detect.*: substitutions declared DIFFERENT. Queries and tokens are averaged over different- and same-model pairs. Values are mean ± std over 5 runs.

SCOUT is both accurate and token-efficient. It reaches a verdict in 3.7 questions and 1.4k tokens on average while keeping its false-positive rate at 2.0%, below the 5% target. RUT has the strongest detection rate among the baselines, but still uses about 105× more tokens. tinyBenchmarks uses 16.6× as many tokens and misses about one fifth of substitutions. These results show that evidence-per-token selection substantially reduces audit cost while retaining high detection.

5.3 Query selection ablation

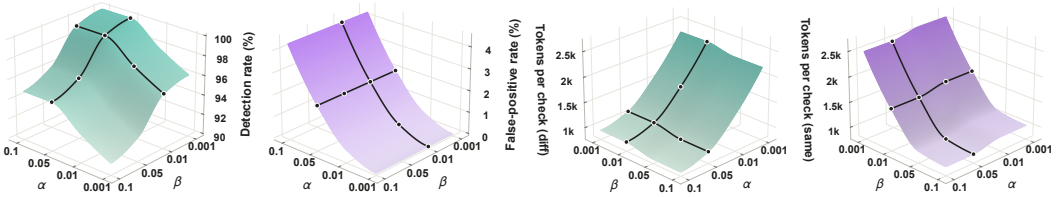
Table 3: Query-selection ablation. *Random questions*: the same sequential test with uniformly sampled questions; *no token pricing*: maximizing G_t rather than $G_t/\hat{c}_q$; *informed prior*: assigning half of the prior mass to the reference's closest benchmark models.

Variant	FPR (%)	Detect. (%)	Queries (diff / same)	Tokens (diff / same)
SCOUT (ours)	**2.0**	**97.2**	**3.0 / 4.3**	**1.1k / 1.6k**
random questions	4.8	90.8	15 / 24	20k / 32k
no token pricing	2.2	98.3	2.7 / 3.7	4.0k / 5.6k
informed prior	1.8	98.5	8.0 /3.0	3.7k / 1.1K
Oracle (substitute known)	–	99.7	2.5 / –	932 / –

Question selection drives the efficiency gain. With random questions, the same sequential test needs about 5× more queries and 18× more tokens, while detecting only 90.8% of substitutes (Table 3). Removing token pricing slightly reduces the number of queries but greatly increases total token use, because it selects highly discriminative but expensive questions. The informed prior reduces the cost of same-model checks but increases the cost of substitution checks, consistent with the benefit of a prior that more closely matches the true substitute.

5.4 Decision threshold sensitivity

Figure 3 shows how the two thresholds trade accuracy against cost. Lowering α reduces false alarms, and the false-positive rate stays below α at every setting, but more tokens are needed to flag a substituted endpoint. Lowering β makes it harder for a substitute to be wrongly accepted as SAME, which raises detection, but more tokens are needed to confirm a genuine endpoint. Each threshold mainly affects one type of error and the cost on one type of endpoint, so the two can be tuned independently. All metrics change gradually across the range tested, and the default setting ($\alpha = 0.05$, $\beta = 0.01$) gives high detection with a low false-positive rate.



(a) Detection rate (b) False-positive rate (c) Tokens (diff) (d) Tokens (same)

Figure 3: Joint effect of α and β on detection, false-positive rate, and token use across same-model and different-model trials.

5.5 Posterior Estimator

We test whether the output reader g_ϕ, a DistilBERT classifier that reads one answer and predicts which of the 4,031 benchmark models produced it, generalizes to questions unseen during training. We ask whether one answer can identify its model and whether this signal reduces audit cost without affecting validity. Writing style contains model-specific information: from a single endpoint answer, the reader ranks the true model first 24.9% of the time and in its top 10 for 53.2% of answers. Enabling the reader raises detection from 96.5% to 97.2% while keeping FPR below 2% and reducing the token cost of both different- and same-model audits. It therefore improves the substitute prediction without changing the validity of the sequential test.

Table 4: Posterior estimator. *Identification*: how often the true model appears in the reader's top-k among 4,031 benchmark models, given one answer to an unseen question; without the reader this is chance. *Audit*: audits with the reader disabled (γ=0) or enabled (γ=1); queries and tokens are for different-model / same-model checks.

	Identification (%)			Audit			
	top-1	top-10	top-100	FPR (%)	Detect. (%)	Queries	Tokens
Without reader (γ=0)	0.025	0.25	2.5	1.6	96.5	2.8 / 4.1	3.2k / 4.8k
With reader (γ=1)	24.9	53.2	84.8	1.9	97.2	2.6 / 4.1	3.0k / 4.7k

5.6 Evaluation on Real Model APIs

We also evaluate SCOUT on 14 models from six families served through Volcengine Ark and OpenRouter. We construct a 400-question benchmark with 100 questions each from GSM8K, MATH, BoolQ, and Social IQa, using fixed answer formats and deterministic grading. Separate profiling responses estimate model success probabilities. Evaluation includes 1,490 audits replayed on held-out API responses and 1,000 live audits with fresh responses. Appendix G provides the full setup and additional results; Table 8 reports profiling and data-collection costs separately from the per-audit costs. As shown in Figure 4, SCOUT detects 96.7% of same-family and 92.9% of other-family substitutions at a 1.5% false-positive rate. It uses only 2.9–3.8 queries and 289–387 tokens per audit, outperforming the compared methods on the accuracy-cost trade-off (Appendix G).

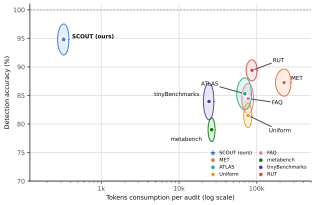


Figure 4: Detection rate versus token cost per audit on real model APIs. Detection is averaged across same-family and other-family substitutions. Ellipses show one standard deviation along each axis. SCOUT achieves high detection at substantially lower token cost.

6 Conclusion

We introduce SCOUT for detecting undisclosed LLM API model substitution when identity checks are unreliable, using Bayesian, token-aware sequential probing with anytime-valid false-alarm control. Across the Open LLM Leaderboard and 14 commercial APIs, SCOUT is more accurate than prior methods while using over 90% fewer tokens and typically deciding after about four questions; future work will address open-ended tasks and adaptive providers.

AI USE STATEMENT

Generative AI tools were used solely as assistive tools for language editing, programming support, and drafting non-substantive text. All research decisions, methodology, experiments, analysis, and final manuscript content were designed, verified, and approved by the authors, who take full responsibility for this work.

ETHICS STATEMENT

SCOUT is intended to help API users audit whether a service matches its advertised model. The method could impose costs on providers or be misused to probe service behavior. We therefore use public benchmark questions and authorized API access, follow provider terms and rate limits, and do not collect personal or sensitive data. SCOUT detects performance discrepancies rather than provider intent or model identity; possible substitutions should be reported with uncertainty and, where appropriate, through responsible disclosure.

REPRODUCIBILITY STATEMENT

We provide the benchmark sources, evaluation protocol, prompts, decoding and grading settings, model profiles, token accounting procedure, and all hyperparameters needed to reproduce our experiments. Upon acceptance, we will release all code and data, including the audit implementation, posterior estimator, configurations, random seeds, and scripts for reproducing every table and figure.

REFERENCES

Tomas Bruckner. One token is enough: Fingerprinting and verifying large language models from single-token output distributions, 2026. URL https://arxiv.org/abs/2607.10252. arXiv:2607.10252.

Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in LLM APIs. *arXiv preprint arXiv:2504.04715*, 2025a.

Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in llm apis. *arXiv preprint arXiv:2504.04715*, 2025b.

A. Philip Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2):278–292, 1984.

Alden Dima, James Foulds, Shimei Pan, and Philip Feldman. You've changed: Detecting modification of black-box large language models. *arXiv preprint arXiv:2504.12335*, 2025.

Werner Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967. doi: 10.1287/mnsc.13.7.492.

Yijia Fang, Yiqing Feng, Bingyu Li, and Mingxun Zhou. KBF: Knowledge boundary as fingerprint for language model and black-box API auditing. *arXiv preprint arXiv:2605.29524*, 2026.

Kehua Feng, Keyan Ding, Hongzhi Tan, Kede Ma, Zhihua Wang, Shuangquan Guo, Yuzhou Cheng, Ge Sun, Guozhou Zheng, Qiang Zhang, and Huajun Chen. Sample-efficient human evaluation of large language models via maximum discrepancy competition. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10913–10947. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.535. URL https://aclanthology.org/2025.acl-long.535/.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open LLM leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this API serving? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d73234a13815fc1f9779dd17d89be9b4-Abstract-Conference.html.

Peter Grünwald and Thijs van Ommen. Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12(4):1069–1103, 2017.

Peter Grünwald, Rianne de Heide, and Wouter M. Koolen. Safe testing. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(5):1091–1128, 2024. doi: 10.1093/jrsssb/qkae011.

Martin Gubri, Dennis Ulmer, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. TRAP: Targeted random adversarial prompt honeypot for black-box identification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11496–11517. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.683. URL https://aclanthology.org/2024.findings-acl.683/.

Valentin Hofmann, David Heineman, Ian Magnusson, Kyle Lo, Jesse Dodge, Maarten Sap, Pang Wei Koh, Chun Wang, Hannaneh Hajishirzi, and Noah A. Smith. Fluid language model benchmarking. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2509.11106.

Yuepeng Hu, Zhengyuan Jiang, Mengyuan Li, Osama Ahmed, Zhicong Huang, Cheng Hong, and Neil Zhenqiang Gong. Fingerprinting LLMs via prompt injection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11795–11810. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.541. URL https://aclanthology.org/2026.acl-long.541/.

Alex Kipnis, Konstantinos Voudouris, Luca Schulze Buschoff, and Eric Schulz. metabench - a sparse benchmark of reasoning and knowledge in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4ebc26584810a189ef1e4f173aba4319-Abstract-Conference.html.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

Peiyu Li, Xiuxiu Tang, Si Chen, Ying Cheng, Ronald Metoyer, Ting Hua, and Nitesh V. Chawla. Adaptive testing for LLM evaluation: A psychometric alternative to static benchmarks, 2026. URL https://arxiv.org/abs/2511.04689v3. arXiv:2511.04689, version 3. First posted in 2025; accepted at ICML 2026.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinyBenchmarks: evaluating LLMs with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34303–34326. PMLR, 2024a. URL https://proceedings.mlr.press/v235/maia-polo24a.html.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: Evaluating LLMs with fewer examples. In *International Conference on Machine Learning (ICML)*, 2024b.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. Efficient multi-prompt evaluation of LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024c. doi: 10.52202/079017-0707. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28236482f64a72eec43706b6f3a6c511-Abstract-Conference.html.

Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. LLMmap: Fingerprinting for large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 299–318. USENIX Association, 2025. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/pasquini.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. Efficient benchmarking (of language models). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2519–2536. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.139. URL https://aclanthology.org/2024.naacl-long.139/.

Mingjie Sun, Yida Yin, Zhiqiu Xu, J. Zico Kolter, and Zhuang Liu. Idiosyncrasies in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57854–57885. PMLR, 2025. URL https://proceedings.mlr.press/v267/sun25z.html.

Sang T. Truong, Yuheng Tu, Percy Liang, Bo Li, and Sanmi Koyejo. Reliable and efficient amortized model-based evaluation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 60238–60265. PMLR, 2025. URL https://proceedings.mlr.press/v267/truong25c.html.

Jean Ville. *Étude critique de la notion de collectif*. Number 3 in Monographies des Probabilités. Gauthier-Villars, Paris, 1939.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16 (2):117–186, 1945. doi: 10.1214/aoms/1177731118.

Skyler Wu, Yash Nair, and Emmanuel J. Candès. Efficient evaluation of LLM performance with statistical guarantees, 2026. URL https://arxiv.org/abs/2601.20251v3. arXiv:2601.20251, version 3.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3277–3306. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.180. URL https://aclanthology.org/2024.naacl-long.180/.

Jihan Yao, Gantavya Bhatt, Arnav Das, Peter Jin, Ke Bao, Qiaolin Yu, Khushi Bhardwaj, Chang Su, Jialei Wang, Yikai Zhu, Sugam Devare, Damon Mosk-Aoyama, Zhen Dong, Venkat Krishna Srinivasan, Yineng Zhang, Oleksii Kuchaiev, Jiantao Jiao, Banghua Zhu, and Jeff Bilmes. Coresets before score sets: Evaluation-unsupervised prompt subset selection for LLM benchmarks, 2026. URL https://arxiv.org/abs/2607.09739. arXiv:2607.09739.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. REEF: Representation encoding fingerprints for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/77e830bd802f4b63b682038d090a97b6-Abstract-Conference.html.

Xiaoyuan Zhu, Yaowen Ye, Tianyi Qiu, Hanlin Zhu, Sijun Tan, Ajraf Mannan, Jonathan Michala, Raluca Ada Popa, and Willie Neiswanger. Auditing black-box LLM APIs with a rank-based uniformity test, 2026. URL https://arxiv.org/abs/2506.06975v5. arXiv:2506.06975, version 5. First posted in 2025.

A IDENTITY DETECTION METHODS

This section describes the signals and reference access behind the five methods introduced in Section 3.1. The first four are the baselines evaluated in Figure 1; output-distribution comparison is discussed using results from prior work and is not included among those four AUC panels.

Direct self-identification. The auditor asks the API endpoint to state its model name, version, or tier and compares the answer with the advertised identity. The check needs no reference samples and can be performed with one query. Its evidence is only the model's generated description of itself: the response may name a vendor or family without specifying a tier, may reflect an earlier model name, or may follow a system instruction about what to say (Cai et al., 2025a). For instance, in our evaluation, different Claude 4.5 tiers gave the same self-reported identity. Such an answer cannot by itself authenticate the backend model.

Canary fingerprinting. The auditor first registers responses from the authentic model to a fixed set of diagnostic prompts. Later, the auditor sends the same prompts to the endpoint and compares its answers with the registered responses. Our experiment used 24 canary probes. This is a simple instance of behavioral fingerprinting; other methods use selected queries or adversarial prompts to elicit more distinctive answers (Gubri et al., 2024; Pasquini et al., 2025). The fingerprint is useful only if the authentic model reproduces its own registered answers and substitutes respond differently. In our measurements, the authentic model reproduced a registered answer only 60% of the time even at temperature zero, while related models sometimes gave identical answers to the same probes.

Watermark verification. A keyed watermark alters the sampling process so that generated text contains a statistically detectable token pattern (Kirchenbauer et al., 2023). The auditor collects text from the endpoint and applies the corresponding keyed detector, which requires the watermark key and tokenizer. This procedure tests for the configured generation-time mark. If a provider uses the same watermark configuration for several tiers, their outputs can carry the same mark regardless of which model produced them. Conversely, the absence of a detected mark is not informative about model identity without knowing that the advertised endpoint was configured to emit that mark.

Trained discriminator. For each proposed reference–substitute pair, the auditor gathers labelled responses from both models and fits a classifier to distinguish their outputs. Our baseline uses 328 labelled responses in total per pair; its score on a new response provides the discrimination signal measured by AUC. Unlike the preceding checks, this procedure is given examples from the candidate substitute before evaluation. If the replacement model is unknown, the auditor must obtain examples of plausible candidates and fit a separate pairwise discriminator for each. A change in either endpoint's response behavior can also make its previously learned decision rule stale.

A EVIDENCE PER QUERY AND THE TOKEN PRICE

Throughout the appendix, $\mathcal{F}_t$ denotes the history of a check before step t: the questions $q_0, \dots, q_{t-1}$, their outputs $o_0, \dots, o_{t-1}$, and any internal randomness of the auditor. The question q_t, the weights $w_{\cdot,t}$, and the prediction $P_{C,t}$ are functions of $\mathcal{F}_t$. We assume that the endpoint answers every query afresh: given $\mathcal{F}_t$ and q_t, $y_t \sim \mathrm{Bern}\big(p_C(q_t)\big)$.

Lemma 1 (Expected evidence per query)**.** *If the endpoint answers q with predicted success probability $P = P_{C,t}(q)$, then*

$$\mathbb{E}[\log e_t \mid \mathcal{F}_t,\, q_t = q] = G_t(q).$$

Proof. Let $\hat{p} = \hat{p}_q$. By Eq. equation 6,

$$\begin{aligned}\mathbb{E}[\log e_t \mid \mathcal{F}_t,\, q_t = q] &= P \log \frac{P}{\hat{p}} + (1-P) \log \frac{1-P}{1-\hat{p}} \\ &= \mathrm{KL}(P \,\|\, \hat{p}) = G_t(q).\end{aligned}$$

□

Proof. Write $\hat{p} = \hat{p}_q$. By Eq. (6),

$$\mathbb{E}\big[\log e_t \mid \mathcal{F}_t,\, q_t = q\big] = P \log \frac{P}{\hat{p}} + (1-P) \log \frac{1-P}{1-\hat{p}} = \mathrm{KL}\big(P \,\|\, \hat{p}\big) = G_t(q). \qquad \square$$

Why evidence per token. Let $M_t = \log E_t - \sum_{u<t} G_u(q_u)$. By Lemma 1, M_t is a martingale when the endpoint behaves as predicted, and its increments are bounded because $\hat{p}_q$ and every $p_m(q)$, hence every $P_{C,t}(q)$, lie strictly inside $(0, 1)$. For a check that stops at a time τ with $\mathbb{E}[\tau] < \infty$, optional stopping gives

$$\mathbb{E}\big[\log E_\tau\big] = \mathbb{E}\Big[\sum_{t<\tau} G_t(q_t)\Big]. \tag{7}$$

A DIFFERENT decision is reached when $\log E_t$ first exceeds $\log(1/\alpha)$, so, ignoring the overshoot, the evidence collected is about $\log(1/\alpha)$. If every asked question yields evidence at the same rate $r = G_t(q_t)/\hat{c}_{q_t}$ per token, then $\sum_{t<\tau} G_t(q_t) = r \sum_{t<\tau} \hat{c}_{q_t}$, and Eq. (7) gives an expected cost of about $\log(1/\alpha)/r$ tokens. This is the token analogue of Wald's approximation to the expected sample size of a sequential test (Wald, 1945): the tokens needed are the evidence needed divided by the evidence per token. Choosing at every step the question with the largest ratio $G_t(q)/\hat{c}_q$ is the greedy rule for making this cost small. Because G_t changes as the weights are updated, this is a design principle rather than an optimality guarantee.

Lemma 2 (Dinkelbach's reformulation)**.** *Let $\lambda_t = \max_q G_t(q)/\hat{c}_q$ over the unasked questions. Then $\max_q U_t(q) = 0$, and $U_t(q) = 0$ exactly for the questions that attain the largest ratio $G_t(q)/\hat{c}_q$.*

Proof. For every unasked q, since $\hat{c}_q > 0$, $U_t(q) = G_t(q) - \lambda_t \hat{c}_q = \hat{c}_q\big(G_t(q)/\hat{c}_q - \lambda_t\big) \le 0$, with equality if and only if $G_t(q)/\hat{c}_q = \lambda_t$. A question attaining the maximum ratio exists because the pool is finite. □

This is the optimality condition of Dinkelbach's method for fractional programs (Dinkelbach, 1967): $\lambda^\star$ is the optimal ratio of $N(x)/D(x)$ with $D > 0$ if and only if $\max_x N(x) - \lambda^\star D(x) = 0$. Dinkelbach's algorithm iterates on λ when the ratio cannot be maximized directly; our pool is finite, so λ_t is computed in one pass and no iteration is needed.

B THE OUTPUT READER AND ITS TEMPERING

Reader. The reader g_ϕ is a small language model with a classification head over the benchmark models $\mathcal{M}$. It is trained with the cross-entropy loss on triples (q, o, m), where o is an output that model m produced for question q. If the training set is balanced across models, a calibrated reader satisfies, by Bayes' rule,

$$g_\phi(m \mid q, o) = \frac{\mathbb{P}(o \mid q, m)}{\sum_{m' \in \mathcal{M}} \mathbb{P}(o \mid q, m')}, \tag{8}$$

so it is proportional, as a function of m, to the likelihood of the output text under model m. Multiplying $w_{m,t}$ by $g_\phi(m \mid q_t, o_t)$ is then a Bayes update on the text of the output; the normalization in Eq. (5) absorbs the denominator.

Why temper. The full update ($\gamma = 1$) is too sharp for two reasons. First, correctness is a function of the output, since y_t is parsed from o_t, so the text likelihood $\mathbb{P}(o_t \mid q_t, m)$ already contains $\mathrm{Bern}\big(y_t; p_m(q_t)\big)$, and multiplying by both factors counts correctness twice. Second, the reader is trained on finite data and is miscalibrated in practice, typically overconfident on outputs unlike those it was trained on. Both are forms of model misspecification, and a standard remedy is to raise the misspecified likelihood to a power $\gamma \in [0, 1]$, which gives a tempered (generalized) posterior (Grünwald & van Ommen, 2017). Setting $\gamma = 0$ ignores the reader, and $\gamma = 1$ trusts it fully.

No effect on validity. The reader enters only through $w_{m,t+1}$, which depends on $o_0, \ldots, o_t$ and is therefore a function of $\mathcal{F}_{t+1}$. The prediction $P_{C,t+1}$ thus still depends only on the history before step $t + 1$, as Proposition 1 requires, whatever γ is and however the reader errs. The reader can change how fast E_t grows under a substitute, but not its expectation under H_0.

C PROOF OF PROPOSITION 1

After a check stops we set $e_t = 1$, so E_t is defined for all t and frozen at its final value; this covers every stopping rule, including SAME and the token budget. Under H_0 with $\hat{p}_q = p_R(q)$, the fresh-answer assumption of Appendix A gives $y_t \mid \mathcal{F}_t, q_t \sim \mathrm{Bern}(\hat{p}_{q_t})$.

Proof of Proposition 1. Each factor is a fair bet. Write $P = P_{C,t}(q_t)$ and $\hat{p} = \hat{p}_{q_t}$, both functions of $\mathcal{F}_t$. Since $\hat{p} \in (0, 1)$, e_t is finite, and

$$\mathbb{E}[e_t \mid \mathcal{F}_t] = \hat{p} \cdot \frac{P}{\hat{p}} + (1 - \hat{p}) \cdot \frac{1 - P}{1 - \hat{p}} = P + (1 - P) = 1.$$

E_t is a nonnegative martingale. $E_t = \prod_{u<t} e_u \geq 0$ is a function of $\mathcal{F}_t$, and $\mathbb{E}[E_{t+1} \mid \mathcal{F}_t] = E_t\, \mathbb{E}[e_t \mid \mathcal{F}_t] = E_t$, with $E_0 = 1$.

Ville's inequality. Let $\tau = \inf\{t : E_t \geq 1/\alpha\}$. For every n, $\tau \wedge n$ is a bounded stopping time, so by optional stopping

$$1 = \mathbb{E}\big[E_{\tau \wedge n}\big] \geq \mathbb{E}\big[E_{\tau \wedge n}\, \mathbf{1}\{\tau \leq n\}\big] \geq \frac{1}{\alpha}\, \mathbb{P}(\tau \leq n).$$

Hence $\mathbb{P}(\tau \leq n) \leq \alpha$ for every n, and letting $n \to \infty$ gives $\mathbb{P}\big(E_t \geq 1/\alpha \text{ for some } t\big) = \mathbb{P}(\tau < \infty) \leq \alpha$ (Ville, 1939). □

Since $\mathrm{Conf}_t \geq 1 - \alpha$ if and only if $\max_{s \leq t} E_s \geq 1/\alpha$, the same bound gives $\mathbb{P}\big(\mathrm{Conf}_t \geq 1 - \alpha \text{ for some } t\big) \leq \alpha$ under H_0. The proof uses only that q_t and $P_{C,t}$ are functions of $\mathcal{F}_t$, so it holds for any selection rule, prior, and reader, and at any stopping time; this is the anytime validity of e-processes (Grünwald et al., 2024).

Estimated reference profile. In practice $\hat{p}_q$ only estimates $p_R(q)$. Let $\delta_q = p_R(q) - \hat{p}_q$. Repeating the first step with $y_t \sim \mathrm{Bern}\big(p_R(q_t)\big)$ gives

$$\mathbb{E}[e_t \mid \mathcal{F}_t] = 1 + \delta\Big(\frac{P}{\hat{p}} - \frac{1 - P}{1 - \hat{p}}\Big) = 1 + \eta_t, \qquad \eta_t = \frac{\delta_{q_t}\big(P_{C,t}(q_t) - \hat{p}_{q_t}\big)}{\hat{p}_{q_t}\big(1 - \hat{p}_{q_t}\big)},$$

where $\delta = \delta_{q_t}$. Here η_t is a function of $\mathcal{F}_t$ and $1 + \eta_t > 0$, since it is the expectation of a positive variable. Therefore $\tilde{E}_t = E_t / \prod_{u<t}(1+\eta_u)$ is a nonnegative martingale with $\tilde{E}_0 = 1$. Suppose $|\eta_t| \le \bar{\eta}$ for every asked question and a check asks at most T questions. Then $E_t \ge 1/\alpha$ implies $\tilde{E}_t \ge 1/\big(\alpha(1+\bar{\eta})^T\big)$, and Ville's inequality applied to $\tilde{E}_t$ gives

$$\mathbb{P}_{H_0}\big(E_t \ge 1/\alpha \text{ for some } t\big) \le \alpha\,(1+\bar{\eta})^T. \tag{9}$$

The bound returns to α as the profile becomes exact. Since δ_q is roughly of order $\sqrt{\hat{p}_q(1-\hat{p}_q)/n_q}$, η_t is roughly of order $|P_{C,t}(q_t) - \hat{p}_{q_t}| / \sqrt{n_q\,\hat{p}_{q_t}(1-\hat{p}_{q_t})}$: more profiling runs on the questions that checks actually ask keep the false-alarm probability close to α.

D Posterior probability of the reference

We compare two hypotheses about the endpoint. Under the first, $C = R$, with prior probability π_R, and each answer follows $y_u \sim \mathrm{Bern}(\hat{p}_{q_u})$ (taking $\hat{p}_q = p_R(q)$ as in Proposition 1). Under the second, C is a benchmark model m drawn from the prior $w_{\cdot,0}$, with total prior probability $1 - \pi_R$. In both cases q_u is chosen by the same rule from the history, so its probability is the same under both hypotheses and cancels from the posterior odds:

$$\frac{w_{R,t}}{1-w_{R,t}} = \frac{\pi_R}{1-\pi_R} \cdot \frac{\prod_{u<t} \mathrm{Bern}\big(y_u; \hat{p}_{q_u}\big)}{L_t}, \qquad L_t = \sum_{m \in \mathcal{M}} w_{m,0} \prod_{u<t} \mathrm{Bern}\big(y_u; p_m(q_u)\big). \tag{10}$$

Lemma 3 (Marginal likelihood of the mixture)**.** *If the weights follow Eq. equation 5, then*

$$L_t = \prod_{u<t} \mathrm{Bern}(y_u; P_{C,u}(q_u))\,.$$

Proof. By induction on Eq. (5), $w_{m,u} = w_{m,0} \prod_{v<u} \mathrm{Bern}\big(y_v; p_m(q_v)\big) / L_u$, with $L_0 = 1$. Hence $L_{u+1}/L_u = \sum_m w_{m,u}\,\mathrm{Bern}\big(y_u; p_m(q_u)\big)$. Because $y_u \in \{0,1\}$, $\mathrm{Bern}(y_u; p)$ is affine in p, so

$$\sum_{m \in \mathcal{M}} w_{m,u}\,\mathrm{Bern}\big(y_u; p_m(q_u)\big) = \mathrm{Bern}\Big(y_u; \sum_{m \in \mathcal{M}} w_{m,u}\,p_m(q_u)\Big) = \mathrm{Bern}\big(y_u; P_{C,u}(q_u)\big).$$

Multiplying over $u < t$ gives the claim. □

Substituting Lemma 3 into Eq. (10), the ratio of likelihoods is exactly $1/E_t$ (Eq. (6)), so

$$\frac{w_{R,t}}{1-w_{R,t}} = \frac{\pi_R}{(1-\pi_R)\,E_t}, \qquad w_{R,t} = \frac{\pi_R}{\pi_R + (1-\pi_R)\,E_t}.$$

The SAME rule $w_{R,t} \ge 1-\beta$ is therefore equivalent to $E_t \le \frac{\beta}{1-\beta} \cdot \frac{\pi_R}{1-\pi_R}$, so both stopping rules are thresholds on the same statistic: DIFFERENT above $1/\alpha$ and SAME below $\beta\pi_R / \big((1-\beta)(1-\pi_R)\big)$. Because $E_0 = 1$, the prior must satisfy $\pi_R < 1-\beta$; otherwise a check would stop before asking anything.

With the reader. When the weights also include the reader's factor, the first equality of Lemma 3 no longer holds, but $\prod_{u<t} \mathrm{Bern}\big(y_u; P_{C,u}(q_u)\big)$ is still the probability that the substitute forecaster $P_{C,u}$ assigns to the observed answers, each predicted from the history before it. Treating the substitute hypothesis as this sequential forecaster, as in the prequential approach (Dawid, 1984), the formula for $w_{R,t}$ is unchanged. Unlike DIFFERENT, the SAME conclusion carries no frequentist guarantee: it depends on π_R, on the benchmark models, and on the accuracy of $\hat{p}_q$.

E Additional experiment settings

Stochastic responses. The leaderboard records one deterministic outcome $z_{m,q} \in \{0,1\}$ per model and question, whereas our method treats answers as random. We therefore turn each record into a success probability

$$p_m(q) = \begin{cases} 1 - a\,\bar{c}_m, & z_{m,q} = 1, \\ b\,\bar{c}_m, & z_{m,q} = 0, \end{cases} \tag{11}$$

where $\bar{c}_m$ is the mean correctness of model m. With $a = b = 0.2/\max_m \bar{c}_m \approx 0.283$, recorded successes receive probabilities in $[0.8, 1]$ and recorded failures in $[0, 0.2]$; the maximum is taken over all 4,500 evaluations, using observed-item averages for incomplete records. Each query returns an independent Bernoulli draw with the endpoint model's probability. Within a trial, all methods see the same draw for the same question, so comparisons are paired; draws are independent across repeated trials. Every method receives these probabilities as the reference and candidate profiles, so $\hat{p}_q = p_R(q)$ and the conditions of Proposition 1 hold exactly. These experiments thus simulate response variability from benchmark records rather than querying live APIs.

F Additional evaluations

F.1 Near-identical substitutes

What we test. A provider would most plausibly swap in a sibling that behaves almost like the advertised model. We group the different-model checks by how often the two models' recorded answers disagree. The model similarity is defined as how much portion of results do two models share the same.

Table 5: Detection rate in % (mean queries) by how often the two models' recorded answers disagree. Different-model checks only; baselines FPR-calibrated as in Table 2.

Model Similarity	<5%	5–10%	10–20%	≥20%
SCOUT (ours)	**85.9 (6.5)**	**94.9 (4.7)**	**98.8 (3.3)**	**99.6 (2.8)**
Oracle	100 (2.7)	100 (2.7)	100 (2.6)	99.6 (2.5)
RUT	12.7 (3,648)	56.1 (3,171)	99.7 (896)	100 (165)
MET	11.3 (23)	23.5 (22)	52.4 (16)	85.0 (9.4)
metabench	9.9 (799)	14.3 (785)	51.1 (598)	88.1 (205)
ATLAS	8.5 (470)	10.2 (470)	44.5 (358)	89.2 (131)

The baselines compare aggregate behaviour, so they lose track of the pair as the two models become similar. Below 10% disagreement they detect only 8–24% of substitutions; RUT reaches 56% only after thousands of questions. SCOUT asks the questions on which this particular pair differs. It detects 86% of substitutes that disagree with the reference on fewer than 5% of questions, using 6.5 questions. Its remaining gap to the oracle is concentrated in these hardest pairs and vanishes as pairs become less alike.

G Real-API Evaluation Details

Benchmark and query protocol. We sample 100 questions from each of GSM8K, MATH, BoolQ, and Social IQa using a fixed seed, yielding 400 questions covering arithmetic reasoning, competition mathematics, reading comprehension, and commonsense reasoning. Table 6 lists the source splits, answer formats, and output limits. Responses are graded using deterministic parsers, with `math-verify` used for MATH. All calls use the same system prompt, temperature 0.7, a per-sample seed, and disabled reasoning or "thinking" modes. Mean input-plus-output token costs vary by approximately $5\times$ across datasets, providing heterogeneous query costs for evaluating SCOUT's token-aware selection.

Models and API providers. We evaluate 14 models from six families (Table 7). Volcengine Ark serves the Doubao-Seed 2.0 and DeepSeek V4 models; OpenRouter serves the Mistral, Llama, Gemma, and Qwen3 models. Each family contains at least two models, allowing both same-family and other-family substitutions for every reference.

Profiling and held-out responses. For each model and question, we collect six profiling responses and estimate the success probability as

$$\hat{p}_q = \frac{k_q + 1}{n_q + 2}, \qquad n_q = 6,$$

Table 6: Benchmark composition. Each dataset contributes 100 questions. Mean tokens include prompt and completion tokens, averaged over all models' profiling responses. Output caps are measured in tokens.

Dataset	Source split (pool size)	Task	Required answer	Output cap	Mean tokens
GSM8K	test (1,319)	math word problems	`Answer: <number>`	384	260
MATH	test, levels 3–5 (3,669)	competition mathematics	`\boxed{}`	640	520
BoolQ	validation (3,270)	reading comprehension	`yes` / `no`	4	185
Social IQa	validation (1,954)	commonsense, 3 choices	`A` / `B` / `C`	4	101

Table 7: Models and providers. Ark denotes Volcengine Ark. Accuracy is measured on the 400-question benchmark. Profiling tokens and charges cover six samples per question. Ark does not report per-call prices.

Model	API	Model identifier	Acc. (%)	Profiling Tokens	Profiling USD
Doubao-Seed-2.0-mini	Ark	`doubao-seed-2-0-mini-260428`	83.2	697k	–
Doubao-Seed-2.0-lite	Ark	`doubao-seed-2-0-lite-260428`	86.9	635k	–
DeepSeek-V4-flash	Ark	`deepseek-v4-flash-ga-260731`	84.8	526k	–
DeepSeek-V4-pro	Ark	`deepseek-v4-pro-ga-260813`	84.9	536k	–
Ministral-3B	OpenRouter	`mistralai/ministral-3b-2512`	69.8	619k	$0.060
Ministral-8B	OpenRouter	`mistralai/ministral-8b-2512`	74.5	636k	$0.093
Ministral-14B	OpenRouter	`mistralai/ministral-14b-2512`	70.9	696k	$0.132
Llama-3.1-8B	OpenRouter	`meta-llama/llama-3.1-8b-instruct`	67.2	633k	$0.036
Llama-3.3-70B	OpenRouter	`meta-llama/llama-3.3-70b-instruct`	80.7	669k	$0.238
Gemma-3-12B	OpenRouter	`google/gemma-3-12b-it`	78.9	623k	$0.063
Gemma-3-27B	OpenRouter	`google/gemma-3-27b-it`	80.4	631k	$0.160
Gemma-4-31B	OpenRouter	`google/gemma-4-31b-it`	86.2	694k	$0.159
Qwen3-30B-A3B-2507	OpenRouter	`qwen/qwen3-30b-a3b-instruct-2507`	76.8	687k	$0.138
Qwen3-235B-A22B-2507	OpenRouter	`qwen/qwen3-235b-a22b-2507`	75.0	675k	$0.213

where k_q is the number of correct responses. These profiles are constructed once and reused across audits. We separately collect four additional responses per model–question pair for replayed evaluation.

Profiling and data-collection costs. Table 8 separates initial profiling, deep profiling, held-out response collection, and table construction. Initial profiling consumes 8.96M tokens across all models, with $1.29 in OpenRouter charges, and completes in 42 minutes using parallel requests. Deep profiling adds 1.01M tokens, approximately 11% of the initial profiling cost. Constructing the probability tables and baseline item banks takes 2.2 seconds.

Table 8: Profiling and evaluation-data collection costs from provider usage records. Tokens cover all models; monetary costs cover OpenRouter only. Requests run in parallel, and wall-clock time is the duration of each phase. Held-out response collection is an experimental evaluation cost, separate from reusable profiling.

Phase	Configuration	Calls	Tokens	USD	Total time spent
Profiling	6 samples × all questions	33,599	8.96M	$1.29	42 min
Deep profiling	14 samples × 40 questions/reference	7,840	1.01M	$0.08	3 min
Held-out responses	4 samples × all questions	22,400	5.98M	$0.83	41 min
Table construction	$\hat{p}$ and baseline item banks	–	–	–	2.2 s

Retrospectively, audits query approximately 13 distinct questions per reference. Profiling only such a subset is estimated to require 8.6k tokens per reference, compared with approximately 640k for

the full pool. This estimate assumes that the informative subset is known in advance; the reported experiments use the profiling procedure and costs above.

Replayed audit protocol. We conduct 1,490 replayed audits: 280 without substitution, 400 with a same-family substitute, and 810 with an other-family substitute. Every model serves in turn as the reference. For each query, the candidate response is drawn at random from that model's four held-out responses, and its recorded token count determines the query cost. Methods share the held-out response draws. Baselines that query the reference receive a reference-model held-out response; when the candidate is the reference itself, the two responses are distinct.

SCOUT uses the other 13 models as $\mathcal{M}$, the thresholds used in the preceding experiments, and a 30k-token budget. Query costs are estimated from mean profiling token counts, and the text reader is disabled. Baselines are refitted using the same 13 models' profiles and may query the entire 400-question pool.

SCOUT variants and additional results. We evaluate the plug-in variant using $\hat{p}_q$ and the robust e-value variant using the deep-profiled reference. The latter's false-alarm guarantee depends on the valid reference uncertainty bounds specified in the method. Both variants make 1,489 correct decisions in 1,490 replayed audits, with no observed false alarms (Table 9).

In a separate exclusion experiment, we remove the actual substitute from $\mathcal{M}$. SCOUT still detects 94.0% of same-family and 95.2% of other-family substitutions. Varying $\alpha \in \{0.01, 0.05, 0.1\}$ and $\beta \in \{0.05, 0.01, 0.001\}$ yields an observed false-positive rate of 0% and balanced accuracy of at least 0.998.

Table 9: Additional replayed-audit results. N, S, and O denote no substitution, same-family substitution, and other-family substitution. Query and token counts are means per audit, excluding profiling. Balanced accuracy is rounded to three decimals.

Configuration	FPR (%)	Same-fam. det. (%)	Other-fam. det. (%)	Bal. acc.	Queries N / S / O	Tokens N / S / O
SCOUT, plug-in $\hat{p}$	0.0	100	99.9	1.000	4.8 / 2.5 / 2.7	516 / 254 / 276
SCOUT, robust + deep	0.0	100	99.9	1.000	3.8 / 2.9 / 3.0	387 / 289 / 306
SCOUT, substitute $\notin \mathcal{M}$	0.0	94.0	95.2	0.964	4.8 / 3.8 / 5.2	516 / 409 / 664

The same-family results illustrate the value of question-level comparisons when aggregate accuracies are similar. For example, DeepSeek-V4-flash and DeepSeek-V4-pro differ by only 0.1 percentage points in overall accuracy (Table 7), while question-level success patterns can still provide evidence of a difference. Conversely, the First disagreement rule achieves 100% substitution detection but also a 100% false-positive rate (Table **??**), illustrating why an isolated disagreement is insufficient for a reliable audit.

Live audit protocol and results. We additionally run 1000 audits using fresh API responses. We select one reference per family and evaluate three conditions: no substitution, a randomly selected same-family substitute, and a randomly selected other-family substitute. Each condition is repeated three times for each of the two SCOUT variants, giving $6 \times 3 \times 3 \times 2 = 108$ audits.

All 108 audits reach the correct decision (Table 10). Condition-level means range from 2.6 to 4.7 queries, 271 to 517 tokens, and 2.3 to 5.6 seconds per audit. The full live evaluation consumes approximately 40k tokens, with $0.003 in recorded charges; per-call Ark prices are unavailable. These fresh-response results complement the replayed evaluation, whose audits reuse a finite collection of held-out responses.

Table 10: Live audits using fresh API responses. Each row contains 18 audits: six reference models with three repetitions each. Queries, tokens, and wall-clock time are means per audit. Wall-clock time runs from the first request to the decision; profiling costs are excluded.

SCOUT variant	Endpoint condition	Correct	Queries	Tokens	Time
Plug-in $\hat{p}$	no substitution	18/18	4.7	517	5.6 s
	same-family substitution	18/18	2.7	271	2.5 s
	other-family substitution	18/18	2.6	277	2.3 s
Robust + deep	no substitution	18/18	4.2	448	3.8 s
	same-family substitution	18/18	3.4	349	2.5 s
	other-family substitution	18/18	3.4	380	3.1 s