Statistical Testing of Large Language Model Reasoning

Didier Chételat [1] **Han Zhou** [1 2] **Reza Moravej** [1] **Joseph Cotnareanu** [1 3] **Yingxue Zhang** [1] **Mark Coates** [3]

Abstract

Improvements in the linguistic abilities of Large Language Models have made it increasingly difficult to distinguish between correct or incorrect answers they might provide in response to a query. In response, many so-called confidence measures have been developed that quantify the degree of confidence an LLM has in an answer based on a given reasoning, at little to no additional cost beyond what was needed to generate the answer. Given such a measure, the answer is then presented to the user only if the confidence is above a certain pre-defined threshold. In this work, we argue that such a procedure can be reinterpreted as a statistical test based on a single reasoning sample, and to improve its power, we propose instead to generalize the test to a mean of multiple samples. To keep the additional computation reasonable, we propose a novel procedure called StepBootstrap which generates alternative reasoning paths at low cost by resampling steps. We show that its confidence measures capture well the ideal distribution, from which we derive a more powerful test while keeping the computational overhead low.

1. Introduction

In recent years, both the general public and domain experts have increasingly relied on Large Language Models (LLMs) to answer questions on topics as varied as healthcare [Gencer and Gencer, 2025], engineering [Nguyen and Kittur, 2025], law [Ariai et al., 2025], and finance [Nie et al., 2024]. To improve answer accuracy, LLMs are typically prompted to produce step-by-step reasoning chains, also known as chains-of-thought, before committing to an answer – a procedure which has been repeatedly demonstrated to be effective [Wei et al., 2022, Li et al., 2025]. However, a confident-sounding reasoning chain may be entirely wrong; the confidence an LLM expresses in an answer and the correctness of that answer are not reliably coupled [Shen et al., 2025]. This is especially concerning given that recent surveys suggest at least two-thirds of users no longer verify the accuracy of LLM-generated responses, a figure expected to keep growing [Ryser et al., 2025].

To address this, a line of work has developed confidence-based filtering methods: the LLM's confidence in its answer is measured, and the answer is presented to the user only if this confidence exceeds a pre-selected threshold. These methods operate at little or no additional computational cost beyond the original generation, since they require no additional autoregressive sampling. Popular confidence measures include answer perplexity [Manakul et al., 2023], semantic entropy [Kuhn et al., 2023], indirect logit scoring [Lin et al., 2022], and verbalized confidence [Jang et al., 2025] — the latter asking the model to explicitly state its confidence as a probability. While these methods differ in how they extract a confidence signal, they share a common structure: a single confidence score is computed from a single reasoning chain, and compared against a threshold [Liu et al., 2025]. The threshold itself is typically either tuned on a validation set or fixed at an arbitrary value such as 50%, with limited theoretical justification.

We observe that this procedure can be reinterpreted as an implicit statistical hypothesis test over a single reasoning chain. Specifically, upon observing an answer and its associated reasoning chain, one is implicitly testing the null hypothesis H_0 that the answer is incorrect against the alternative hypothesis H_1 that it is correct. The confidence score serves as the test statistic, and the null is rejected (that is, the answer is accepted as correct when this statistic exceeds the chosen threshold). To our knowledge, this reinterpretation is novel, and it is productive: it immediately suggests a principled path to improvement. Just as any statistical test becomes more powerful with more informative test statistics, confidence filtering can be improved by basing the test statistic on a richer set of observations than a single reasoning chain.

The natural improvement, then, is to average confidence scores across multiple distinct reasoning chains that reach the same answer, obtaining a more stable and reliable test statistic. Multiple authors have shown that small changes in the language or format of a reasoning chain — let alone

[1]Huawei Noah's Ark Lab, Montreal, Canada [2]University of Waterloo, Waterloo, Canada [3]McGill University, Montreal, Canada. Correspondence to: Didier Chételat <didier.chetelat@huawei.com>.
ICML 2026 Workshop on Hypothesis Testing, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

genuinely different arguments — can dramatically alter confidence measures [Yang et al., 2024a, Liu and Chu, 2026, Wang et al., 2026], suggesting that any single chain may be an unreliable basis for a decision. However, generating alternative reasoning chains for a given answer requires full autoregressive generation and is therefore computationally prohibitive. At that cost, alternative strategies such as Self-Consistency [Wang et al., 2022], which generate multiple complete answers and aggregate them can be utilized.

In this work, we propose instead a computationally feasible alternative. Our strategy, which we call StepBootstrap, involves building alternative chains-of-thought by resampling reasoning steps, in a way reminiscent of the bootstrap method in statistics. We retain the original ordering of the steps and the original conclusion, with steps sampled multiple times appearing one after the other. To make sure it reaches the same conclusion as the original, the last step is always kept untouched. Figure 1 illustrates the construction.

We validate StepBootstrap through extensive experiments on question answering, coding, and mathematical reasoning benchmarks, across four confidence metrics: answer perplexity, semantic entropy, indirect logit scoring, and verbalized confidence. We first show that the mean confidence over step-bootstrapped chains approximates the ideal mean obtained by averaging over fully generated alternative reasoning chains. We then show that this ideal test yields a more powerful hypothesis test than the vanilla single-chain baseline. Finally, we show that StepBootstrap, despite its much lower computational cost, closely tracks this ideal improvement in practice across all four confidence metrics and various benchmark categories.

In summary, our contributions are as follows.

1. We reinterpret previous work on confidence filtering as implicitly performing a statistical test over a single reasoning chain reaching a target answer, and suggest a computationally expensive improvement involving averaging over several reasoning chains reaching that same answer.

2. We propose a practical approximation that still yields improvement in power at minimal additional computational cost, based on generating alternative reasoning chains through step resampling and computing their confidences using forward passes of the LLM.

3. We demonstrate the validity of the approximation and the power improvements through extensive experiments.

2. Methodology

We consider the following setup. In response to a query q, a Large Language Model (LLM) is prompted through a sentence such as "`Let's think step-by-step`" [Kojima et al., 2022] to produce a reasoning chain r, that is a structured sequence of numbered reasoning steps followed by an answer. More precisely, we assume the prompt instructs the LLM to number its steps in a predictable format (e.g. Step 1, Step 2, and so on), and to conclude with a fixed answer extraction formula such as "`The answer is \boxed{<answer>}`", from which the answer answer(r) can be parsed automatically. For example, in response to the query q ="`Which country is more populous, France or Spain?`", the LLM might produce the reasoning chain $r =$

```
"Step 1:  The population of France is
approximately 68 million.
Step 2:  The population of Spain is
approximately 47 million.
Step 3:  Since 68 million > 47 million,
France is more populous.
The answer is \boxed{France}."
```

from which answer(r) ="`France`" would be extracted. We assume the LLM generates reasoning chains by sampling at a fixed temperature, and denote the resulting distribution over reasoning chains produced in response to query q by $p_{R|Q}(r|q)$.

We further assume access to a confidence measure $c_a(r) \in [0, 1]$: a deterministic procedure that quantifies the LLM's confidence that answer $a =$ answer(r) is correct, given reasoning chain r, at little to no additional computational cost beyond generating r. The confidence measures we consider in this work are answer perplexity [Manakul et al., 2023], semantic entropy [Kuhn et al., 2023], indirect logit scoring [Lin et al., 2022], and verbalized confidence [Jang et al., 2025].

2.1. Confidence filtering as hypothesis testing

Given a confidence measure $c_a(r)$, previous authors have proposed filtering LLM answers by only presenting answer(r) to the user if $c_a(r)$ exceeds a pre-selected threshold τ. We observe that this procedure can be reinterpreted as a statistical hypothesis test. Upon observing a reasoning chain r and its associated answer $a =$ answer(r), one is implicitly testing:

$$H_0 : a \text{ is incorrect} \qquad \text{vs} \qquad H_1 : a \text{ is correct}, \quad (1)$$

where $c_a(r)$ serves as the test statistic and τ as the critical value. The null hypothesis H_0 is rejected (the answer is accepted as correct) when $c_a(r) > \tau$. When the null is not rejected, the system may either resample a new reasoning chain or inform the user that no sufficiently confident answer was found.

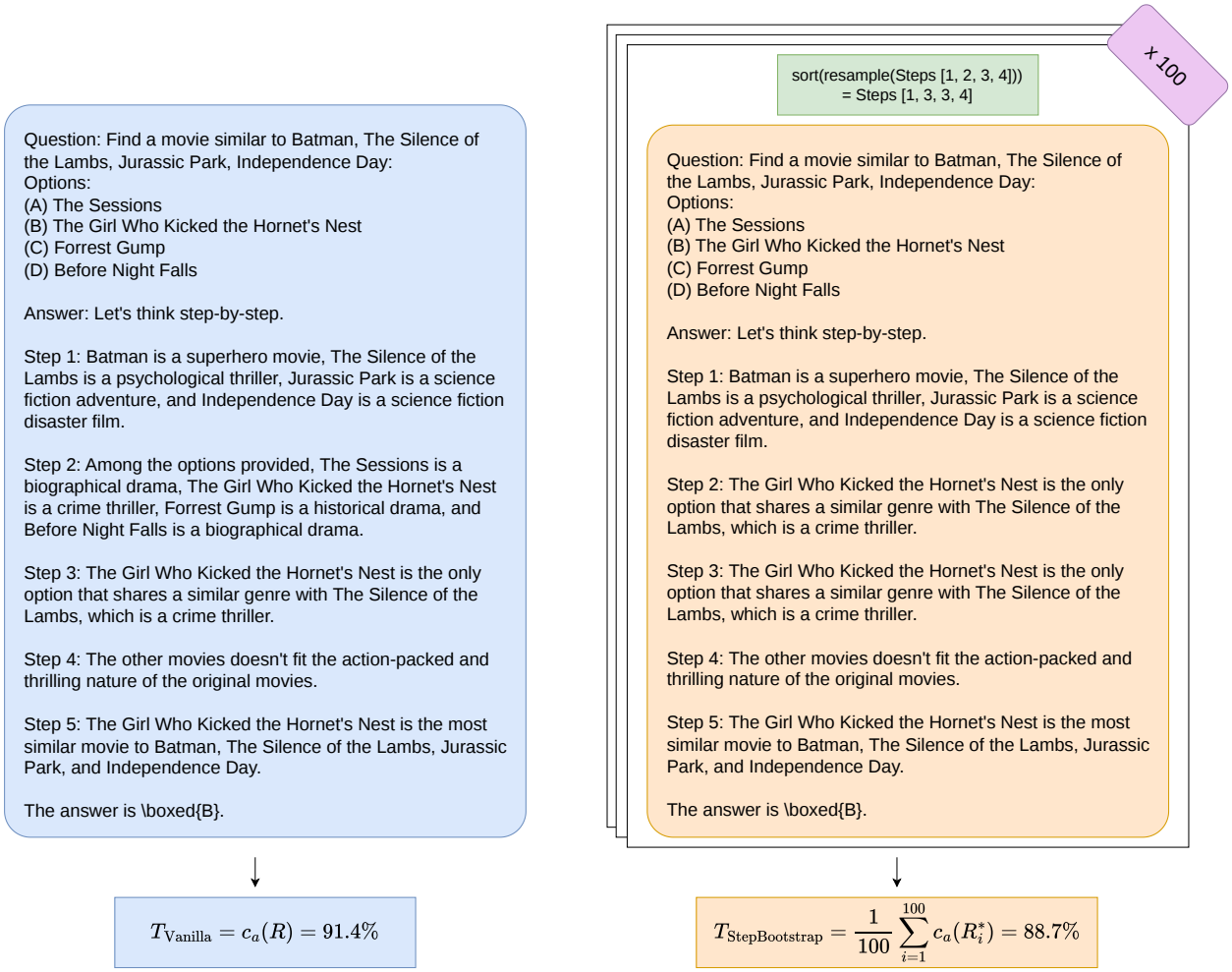


Figure 1. Computation of the test statistics. On the left, the standard procedure where a reasoning chain is generated, and its confidence, which is used as test statistic, is computed using a standard method, such as answer token perplexity or verbalized confidence. On the right, our proposed StepBootstrap method: we resample steps, keeping the original ordering, conclusion and last step. Our test statistic is the average of the confidences over the resampled reasoning chains.

Although this procedure improves the reliability of AI systems, it has been noticed by several authors [Yang et al., 2024a, Liu and Chu, 2026, Wang et al., 2026] that confidence measures can often vary dramatically based on the reasoning. Instead, one could base the test statistic on the average confidence across multiple independent reasoning chains reaching the same answer. By the law of large numbers, such an average would have a lower variance than any individual confidence score, yielding a more stable and reliable basis for the accept/reject decision.

2.2. An improved test

We now formalize this strategy. For a query q and a candidate answer a, consider the conditional distribution over reasoning chains that reach a:

$$p_{R|A,Q}(r|a,q) = \frac{p_{R|Q}(r|q)\ \mathbb{1}[\text{answer}(r) = a]}{\sum_{r'} p_{R|Q}(r'|q)\ \mathbb{1}[\text{answer}(r') = a]}. \quad (2)$$

Given n independent samples $R_1, \ldots, R_n \sim p_{R\,|\,A,Q}(\cdot\,|\,a,q)$, we define the ideal test statistic as the mean confidence:

$$T_{\text{Ideal}} = \frac{1}{n}\sum_{i=1}^{n} c_a(R_i), \quad (3)$$

and reject H_0 – accepting the answer – if $T_{\text{Ideal}} > \tau$. We refer to this as the *ideal test*. As argued above, T_{Ideal} has lower variance than the single-chain statistic $c_a(r)$ and better separates correct from incorrect answers. Unfortunately, the ideal test is computationally prohibitive. Generating n alternative reasoning chains from $p_{R|A,Q}$ requires at least n

Algorithm 1 StepBootstrap confidence sampling.

Input: reasoning chain r, answer a, nb of resamples n
Extract steps $r_1, \ldots, r_K$ from r
for $i \leftarrow 1$ to n **do**
  // Sample with replacement, preserve order
  $I \leftarrow \text{sort}\big(\text{resample}([1, \ldots, K-1],\ \text{size} = K-1)\big)$
  // Fix final step to preserve answer
  $R_i^* \leftarrow \text{reasoning-chain}(r_{I_1}, \ldots, r_{I_{K-1}}, r_K)$
  // No autoregressive generation needed
  Compute $c_a(R_i^*)$ via a forward pass of the LLM
end for
return $c_a(R_1^*), \ldots, c_a(R_n^*)$

full autoregressive generation passes, each comparable in cost to generating the original chain. This defeats the goal of confidence filtering as a lightweight failsafe.

2.3. Our approach: StepBootstrap

We propose instead the following method, which we call StepBootstrap. This approach creates synthetic reasoning chains, by sampling steps with replacement, from which the confidence measures can be computed. As the reasoning chains are generated through text manipulation rather than autoregressive generation, computing their confidence only requires a forward pass by the LLM. This provides a cheap approximation to computing confidences of samples from the ideal distribution given in Eq. 2.

In detail, given an original reasoning chain r with steps $r_1, \ldots, r_K$, StepBootstrap produces n alternative chains as follows. For each $i = 1, \ldots, n$, we sample $K-1$ indices with replacement from $\{1, \ldots, K-1\}$, sort them to preserve the original relative ordering of steps, and construct a new reasoning chain by appending the resampled steps to the fixed final step r_K:

$$R_i^* = \text{reasoning-chain}(r_{I_1}, \ldots, r_{I_{K-1}}, r_K), \text{ where}$$
$$I = \text{sort}\big(\text{resample}([1, \ldots, K-1], \text{size} = K-1)\big).$$

We experimented with taking a smaller number of steps, but we obtained the best results by keeping the total number of steps equal to K. Given such a R_i^*, the confidence $c_a(R_i^*)$ is then evaluated using a single forward pass. Algorithm 1 presents the full procedure.

Finally, given the n bootstrapped confidence scores, we define the StepBootstrap test statistic as their mean:

$$T_{\text{StepBootstrap}} = \frac{1}{n} \sum_{i=1}^{n} c_a(R_i^*). \tag{4}$$

As with the ideal test, we reject H_0 and accept answer a if $T_{\text{StepBootstrap}} > \tau$. Otherwise, like in the original setting, we can resample a new reasoning chain, or inform the user that no sufficiently confident answer was found. Note that if we resample a chain, the StepBootstrap confidences should not be used in another batch test, even if the new chain reaches the same answer.

We show in Section 3.1 that although the step-bootstrapped reasoning chains only approximate the ideal distribution given in Eq. 2, this test statistic $T_{\text{StepBootstrap}}$ approximate well the statistics T_{Ideal} of the ideal test.

3. Experiments

We evaluate on eight datasets: MATH500[1] [Hendrycks et al., 2021], BigBench Movie Recommendation (MOVIEREC[2]) and BigBench Causal Judgment (CAUSAL[2]) [Srivastava et al., 2022], LOGIQA[2] [Liu et al., 2020], HOTPOTQA[2] [Yang et al., 2018], BFCL[2] [Patil et al., 2025], CS1QA[1] [Changyoon Lee and Oh, 2021] and CODEQA[1] [Liu and Wan, 2021]. To keep experimental costs reasonable, we sampled 500 questions from each dataset. Each dataset comes with its own official procedure for determining that two answers are the same: we follow it as much as possible. We use SymPy [Meurer et al., 2017] to evaluate MATH500. MOVIEREC, CAUSAL and LOGIQA are multiple-choice, so we use exact string matching. BFCL comes with an official evaluation function. For HOTPOTQA, CS1QA and CODEQA we used GPT-5-mini[3] [OpenAI, 2025] as judge.

We evaluate on two large language models, Llama 3.1 8B Instruct[4] and Qwen 3.5 27B[2]. An example prompt is provided in Appendix A. All prompts are zero-shot and encourage the model to both number its reasoning steps and summarize its final answer with the answer extraction sentence "`The answer is \boxed{<answer>}`". For Qwen, the model tended to meta-think about the optimal way to number steps between the `<think>` and `</think>` tokens, before producing a final chain-of-thought with numbered steps and the final answer: we let it generate freely within those thinking tokens, but discarded the thinking block before computing the confidence measures.

We generated the data as follows. For every question q in each dataset, we generate a reference reasoning chain r and answer a by sampling with temperature 0.9, and collect its associated confidence measure $c_a(r)$. To generate reasoning chains from $p_{R|A,Q}$, we perform rejection sampling: namely, we sampled 31 (for Llama) and 7 (for Qwen) additional reasoning chains, and only kept those that reached the same answer as the reference chain. The StepBootstrap confidences are generated from the reference sample using Algorithm 1 with $n = 100$ samples.

We report the following confidence measures.

[1] MIT License [2] Apache 2.0 License [3] Licensed under OpenAI's Terms of Use [4] Meta Llama Community License

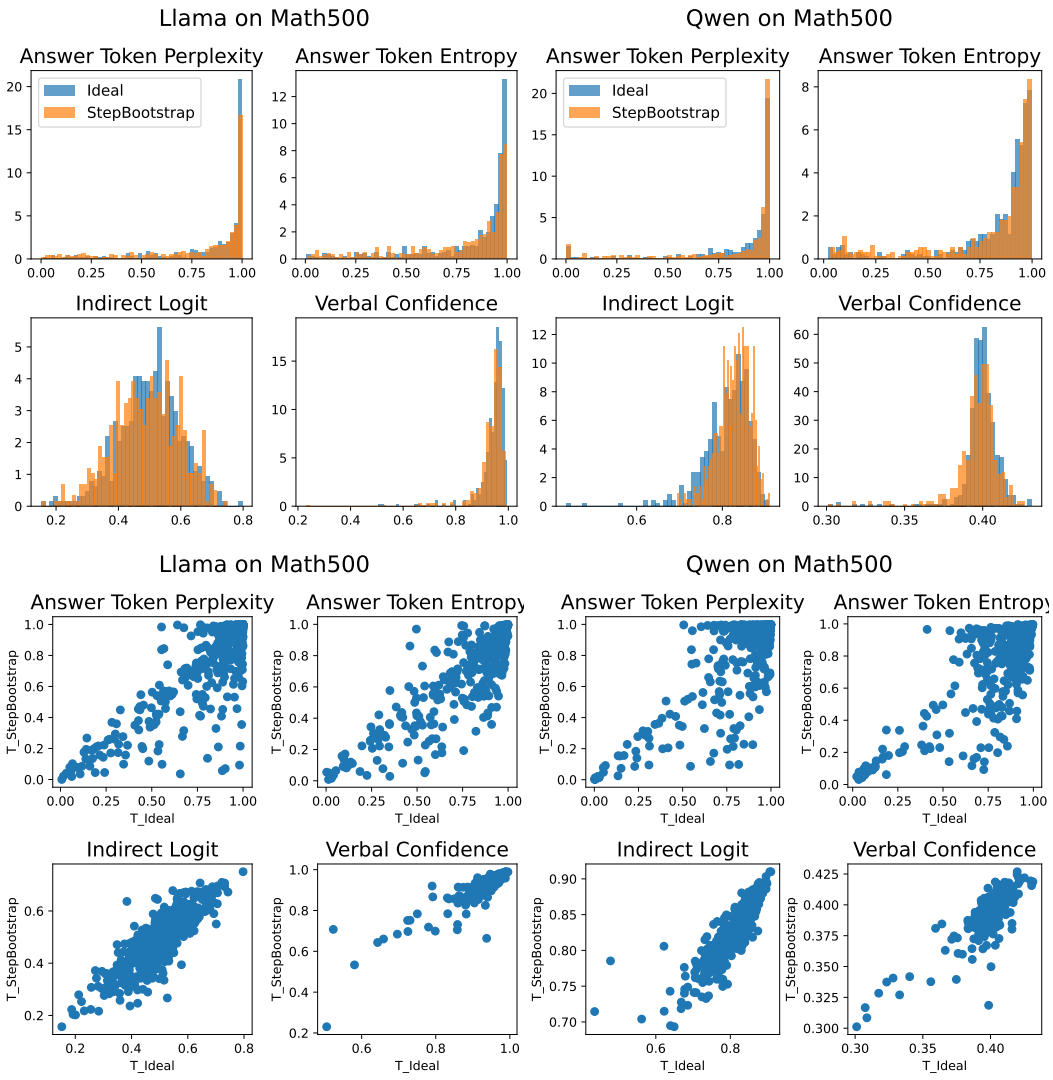


Figure 2. Histograms and scatter plots of the true confidence statistics T_{Ideal} against our approximative $T_{\text{StepBootstrap}}$, for the MATH500 dataset. Results are presented for Llama 3.1 8B and Qwen 3.5 27B and each confidence measure. Plots for the other datasets are available in Appendix C.

Answer token perplexity [Manakul et al., 2023]
The average probability of the tokens $t_1, \ldots, t_m$ appearing within the `\boxed{<answer>}` element of the answer extraction sentence. That is, our measure is $c_a(r) = [\prod_{i=1}^{m} p_{\text{LLM}}(t_i|t_{<i})]^{1/m} \in [0, 1]$.

Answer token entropy [Kuhn et al., 2023]
The average entropy of the distributions from which these tokens were selected. We transform by $c \to \exp(-c)$ to ensure it increases with higher confidence and maps to $[0, 1]$, like the other measures. That is, our measure is $c_a(r) = \exp(-\frac{1}{m}\sum_{i=1}^{m} \text{entropy}[p_{\text{LLM}}(\cdot|t_{<i})]) \in [0, 1]$.

Indirect logit [Lin et al., 2022]
Following the answer extraction sentence "`The answer is \boxed{<answer>}`", we append the string "`\nTrue/False:`", yielding an extended prompt q_{indirect}, and then weight the probability the LLM assigns to "` True`" vs. "` False`". That is, our measure is $c_a(r) = p_{\text{LLM}}(\text{“ True”} \,|\, q_{\text{indirect}})/[p_{\text{LLM}}(\text{“ True”} \,|\, q_{\text{indirect}}) + p_{\text{LLM}}(\text{“ False”} \,|\, q_{\text{indirect}})] \in [0, 1]$.

Verbalized confidence [Jang et al., 2025]
Following the answer extraction sentence "`The answer is \boxed{<answer>}`", we append the string "`Please respond with a score from 0 to 100 in <confidence> </confidence> tags.\nHow confident are you in your previous answer?\n<confidence>`", yielding an extended prompt $q_{\text{verbalized}}$. We then record the probability the LLM assigns to every number from 0 to 100, and compute the weighted average of those numbers, normalized by

Table 1. The Pearson correlation between T_{Ideal} and $T_{\text{StepBootstrap}}$. We report results on every dataset and confidence measure, for Llama 3.1 8B and Qwen 3.5 27B.

	Llama 3.1 8B				Qwen 3.5 27B			
Dataset	**Token Perp**	**Token Ent**	**Indir. Logit**	**Verb. Conf**	**Token Perp**	**Token Ent**	**Indir. Logit**	**Verb. Conf**
MATH500	0.86	0.88	0.85	0.90	0.85	0.82	0.86	0.81
MOVIEREC	0.45	0.39	0.79	0.81	0.78	0.50	0.80	0.72
CAUSAL	0.59	0.59	0.90	0.78	0.70	0.60	0.92	0.64
LOGIQA	0.67	0.68	0.80	0.67	0.43	0.53	0.79	0.83
HOTPOTQA	0.92	0.97	0.86	0.92	0.93	0.95	0.94	0.72
BFCL	0.94	1.0	0.86	0.84	0.99	1.0	0.89	0.78
CS1QA	0.94	0.98	0.91	0.94	0.94	0.98	0.94	0.94
CODEQA	0.08	0.40	0.59	0.83	0.55	0.78	0.64	-0.01

100 to ensure it is on the same scale as the other measures. That is, our measure is $c_a(r) = [0 \cdot p_{\text{LLM}}(\text{“0”} \mid q_{\text{verbalized}}) + \cdots + 100 \cdot p_{\text{LLM}}(\text{“100”} \mid q_{\text{verbalized}})]/100 \in [0, 1]$.

The experiments were run on 8 modern parallel computing devices with 80GB of memory each. Running all the Llama experiments took 33 hours, and running all the Qwen experiments took 502 hours. Code and scripts to reproduce all the experiments is available at the following location.

3.1. Quality of the StepBootstrap approximation

Our method relies on the assumption that confidences over step-bootstrapped reasoning chains accurately approximate the confidences over actually generated reasoning chains. To support this claim, we illustrate the relationship between $T_{\text{StepBootstrap}}$ and T_{Ideal} in a scatter plot, as well as their empirical distributions over questions in benchmark datasets. Figure 2 shows the resulting plots for the MATH500 dataset. Appendix C provide the plots for other datasets.

We also measure the strength of the linear relationship between $T_{\text{StepBootstrap}}$ and T_{Ideal} by computing their Pearson correlation. Table 1 reports $\text{corr}(T_{\text{StepBootstrap}}, T_{\text{Ideal}})$ across all queries in each dataset, where $\text{corr}(\cdot, \cdot)$ stands for the Pearson correlation. Frequent large correlation values indicates a strong linear relationship between $T_{\text{StepBootstrap}}$ and T_{Ideal} across various datasets and confidence estimation methods.

3.2. Main results

We next show that our StepBootstrap procedure allows us to construct a test with better power at various possible thresholds than the standard vanilla test, while having a limited additional computational cost. We use the same data as in the previous section, and compare the following test statistics.

Vanilla We report the confidence of the reference CoT as statistic, $T_{\text{Vanilla}} = c_a(r)$.

StepBootstrap We report the average of $n = 100$ StepBootstrap confidences as statistic, that is $T_{\text{StepBootstrap}} = \sum_{i=1}^{100} c_a(R_i^*)/100$.

Ideal We report the average of the confidences sampled from $p_{R|A,Q}(r|a, q)$ using rejection sampling, that is, $T_{\text{ideal}} = \sum_{i=1}^{n} \mathbb{1}[\text{answer}(R_i) = a] c_a(R_i) / \sum_{i=1}^{n} \mathbb{1}[\text{answer}(R_i) = a]$. We include the original reference confidence, so this sum ranges over $n = 32$ (for Llama) and $n = 8$ (for Qwen) and always includes at least one sample (the reference one).

We report the following evaluation metrics.

AUROC For a given test statistic T, its Area Under the Receiver Operating Characteristic is defined as $\text{AUROC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(u))\text{d}u$ for $\text{TPR}(\tau) = \text{P}[T > \tau \mid \text{correct}(a)]$ and $\text{FPR}(\tau) = \text{P}[T > \tau \mid \text{incorrect}(a)]$. This measures the ability of the statistic to distinguish between correct and incorrect answers over all possible thresholds. The AUROC corresponds to the area under its associated ROC curve, which are provided for reference in Appendix B.

Timing We also report the amount of time required to produce the test statistic, would this method be used as a confidence filtering procedure. This includes the time required to generate the original reference reasoning chain. Thus, for StepBootstrap, the timing represents the amount of time required to produce the reference chain plus $n = 100$ step-bootstraps; for Ideal, this represents the amount of time required to sample $n = 32$ (for Llama) or $n = 8$ (for Qwen) reasoning chains. For ease of reading, in the tables we report ratios with respect to the vanilla timing.

Table 2 provide the main results for Llama and Qwen, re-

Table 2. Main results on Llama 3.1 8B and Qwen 3.5 27B across all datasets. We report the AUROC of each method, averaged over all four confidence metrics, with its bootstrapped standard deviation, in percentage. We bold the largest value of Vanilla or StepBootstrap, respectively. An asterisk (*) denotes that a one-sided Wilcoxon signed-rank test determined that the difference was significantly greater than zero at a 0.05 significance level. In case it was not, we bolded both values. Finally, we also report the average time taken to run each method, per question on each dataset, as a ratio of the vanilla time. Detailed tables per confidence measure can be found in Appendix D.

		Llama 3.1 8B		**Qwen 3.5 27B**	
Dataset	Method	AUROC	Time R.	AUROC	Time R.
MATH500	Vanilla	0.7130 ± 3.8%	1.00	**0.8870 ± 3.9%**	1.00
	StepBootstrap	⋆**0.7362 ± 3.7%**	1.33	**0.8807 ± 4.0%**	1.20
	Ideal	0.7463 ± 3.5%	31.40	0.9366 ± 3.0%	8.10
MOVIEREC	Vanilla	0.6280 ± 5.7%	1.00	0.5345 ± 10.8%	1.00
	StepBootstrap	⋆**0.6456 ± 5.4%**	1.38	⋆**0.5648 ± 9.6%**	1.42
	Ideal	0.6722 ± 5.2%	31.88	0.5628 ± 12.2%	7.92
CAUSAL	Vanilla	0.5378 ± 8.0%	1.00	0.5473 ± 9.0%	1.00
	StepBootstrap	⋆**0.5545 ± 7.7%**	1.43	⋆**0.5557 ± 8.8%**	1.27
	Ideal	0.5532 ± 7.7%	31.71	0.5500 ± 8.9%	7.80
LOGIQA	Vanilla	0.5710 ± 3.9%	1.00	0.5583 ± 5.4%	1.00
	StepBootstrap	⋆**0.6373 ± 4.1%**	1.44	⋆**0.5889 ± 6.2%**	1.30
	Ideal	0.6545 ± 3.7%	30.78	0.5587 ± 6.6%	7.98
HOTPOTQA	Vanilla	0.6530 ± 4.4%	1.00	0.5523 ± 6.5%	1.00
	StepBootstrap	⋆**0.6695 ± 4.2%**	2.00	⋆**0.5610 ± 6.3%**	1.35
	Ideal	0.6662 ± 4.2%	35.63	0.5410 ± 6.2%	8.05
BFCL	Vanilla	**0.5168 ± 10.7%**	1.00	**0.6563 ± 5.3%**	1.00
	StepBootstrap	**0.5161 ± 11.1%**	2.14	**0.6598 ± 5.3%**	3.29
	Ideal	0.5153 ± 11.2%	33.57	0.6430 ± 5.3%	8.05
CS1QA	Vanilla	0.3752 ± 11.7%	1.00	⋆**0.3919 ± 14.3%**	1.00
	StepBootstrap	⋆**0.3808 ± 11.4%**	1.56	0.3768 ± 15.3%	1.44
	Ideal	0.3650 ± 11.9%	34.33	0.3541 ± 15.6%	7.56
CODEQA	Vanilla	0.4594 ± 12.0%	1.00	0.4950 ± 9.3%	1.00
	StepBootstrap	⋆**0.5414 ± 10.2%**	1.75	⋆**0.5113 ± 9.0%**	1.38
	Ideal	0.5097 ± 10.2%	26.00	0.5537 ± 7.4%	7.88

spectively. As can be seen, for both datasets, both StepBootstrap and Ideal provide significant improvements over the Vanilla test in AUROC. However, whereas Ideal does so at greatly increased cost (roughly 32/8 times the cost of Vanilla, as expected), the overhead brought by StepBootstrap is much more reasonable, representing 1.2-3.3x the computational cost of the vanilla test. Appendix D provides further results by dataset, language language model and confidence measure.

4. Related Work

4.1. Confidence estimation of LLM reasoning

There is a large literature on measuring the confidence that a large language model has in a reasoning chain it has generated in response to a query. This is usually referred to as confidence or uncertainty estimation for LLMs. A recent survey [Liu et al., 2025] enumerates 35 methods, although most of these are too expensive for our setting as they either involve generating multiple reasoning chains and scoring them relative to each other, such as in self-consistency [Wang et al., 2023], using an external natural language inference (NLI) model such as Cross-Encoder-Roberta-Large [Duan et al., 2024], or some form of fine-tuning [Liu et al., 2024].

There are, however, multiple confidence evaluation methods that impose limited additional computational overhead. Metrics include the mean probability [Margatina et al., 2023, Manakul et al., 2023, Si et al., 2023, Mora-Cross and Calderon-Ramirez, 2024, Fadeeva et al., 2024], minimum probability [Manakul et al., 2023], and average entropy [Kuhn et al., 2023, Manakul et al., 2023, Fadeeva et al., 2024] of the generated answer tokens. In addition, some authors have proposed to ask a follow-up question like “Is this answer correct?" and weight the probability of the yes/no

tokens. These are known as p(true) or indirect logit methods [Kadavath et al., 2022, Lin et al., 2022]. An alternative is to simply ask the model to output a tokenized confidence score with a followup question like "What is your confidence in this answer between 0 and 100?", a method known as verbalized confidence [Xiong et al., 2024, Lin et al., 2022, Jang et al., 2025]. Finally, a recently introduced metric is DeepConf, which computes token probabilities of a rolling window over the entire reasoning chain, rather than just the answer tokens [Fu et al., 2026].

More generally, this line of work fits within the greater LLM hallucination detection literature, which aims not only to detect whether an answer or a reasoning chain is correct, but more generally any statement uttered by the large language model [Alansari and Luqman, 2026].

4.2. Directed reasoning generation with LLMs

Our StepBootstrap procedure also relates to backwards or "goal-directed" reasoning for LLMs, which aims to create chains of reasoning that reach a certain target answer. All of these methods involve full chain-of-thought generation, rather than the simple forward pass over a pre-constructed chain-of-thought required by our approach.

Most approaches simply involve prompting the LLM to generate reasoning (or a "rationalization") for a question with an answer marked as purportedly correct [Zelikman et al., 2024]. Rajani et al. [2019] train a model to generate answer-conditioned rationales directly. The method of Yang et al. [2024b] rephrases the chains of thought using a "self-distillation template" and rejects any that reach a different answer. Wang et al. [2025] make local edits to a generated reasoning path in order to increase the path's answer perplexity in the model. The aim is to generate a high-quality training dataset for a reasoning model. Self Distillation Fine Tuning RADIO [Jia et al., 2025] uses rephrasing for Retrieval-Augmented Generation. Some algorithms have also been proposed for generating reasoning chains with a directed goal, but in a step-by-step fashion, such as LAMBADA [Kazemi et al., 2023] and Reason From Future [Xu et al., 2025]. Finally, Cetin et al. [2026] train teacher-models to produce rationales for a problem's solution (given in the prompt) in order to train student models by knowledge distillation. Moreover, most do not directly target confidence assessment.

5. Limitations

A primary limitation of this work is no concrete way of selecting a test threshold τ. While this is in line with the majority of the work in the domain, it limits the applicability of the method. Often, statistical tests are designed such that a threshold can be easily set to fix the false positive rate at a target value such as 0.05. A potential extension of this work would be to do the same, by either assuming a parametric distribution or by identifying some transformation of the confidences which makes the threshold obvious.

Furthermore, this method comes with no theoretical guarantees regarding consistency. StepBootstrap introduces a bias, and we show empirically that this bias seems small enough to not prevent gains in recall and precision over various thresholds, but it is not clear whether the bias completely goes to zero as the number of bootstrap samples tends to infinity. Further work in this direction would help clarify the situation.

6. Conclusion

In this work, we provide a novel method for rejecting potentially wrong answers from an LLM in response to a user query, with minimal additional cost beyond the generation of the original answer and associated reasoning. This method involves computing cheap confidence measures on alternative reasoning chains, constructed by sampling with replacement steps from the original reasoning, and using the average confidence as a statistic to accept or reject an answer. Future extensions of this work could investigate more complex statistics of the bootstrapped confidence than the mean, as well as methods for determining a high-quality threshold.

Impact Statement

We believe our work would have positive societal impact, as it aims to reduce the number of misleading answers produced by large language models. Such misleading answers are proving to be problematic, given the observation of Ryser et al. [2025] that many today do not hesitate to believe information produced by LLMs. The main negative impact of our method is that it increases the computational cost of an AI system compared to performing no checks on answers. We note however that this concern is shared with other confidence filtering methods, and we were careful in this work to design an approach whose computational overhead is as limited as possible.

References

Gülcan Gencer and Kerem Gencer. Large language models in healthcare: A bibliometric analysis and examination of research trends. *Journal of Multidisciplinary Healthcare*, pages 223–238, 2025.

San S. Nguyen and Javeed Kittur. Review of current and potential uses of large language models in engineering. In *Frontiers in Education*, volume 10. Frontiers Media SA, 2025.

Farid Ariai, Joel Mackenzie, and Gianluca Demartini. Natural language processing for the legal domain: A survey of tasks, datasets, models, and challenges. *arXiv preprint arXiv:2410.21306*, 2025.

Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges. *arXiv preprint arXiv:2406.11903*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From System 1 to System 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.

Wei Shen, Han Wang, Haoyu Li, and Huan Zhang. Decepchain: Inducing deceptive reasoning in large language models. *arXiv preprint arXiv:2510.00319*, 2025.

Adrian Ryser, Florian Allwein, and Tim Schlippe. Calibrated trust in dealing with LLM hallucinations: A qualitative study. *arXiv preprint arXiv:2512.09088*, 2025.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, December 2023.

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.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022.

Chaeyun Jang, Moonseok Choi, Yegon Kim, Hyungi Lee, and Juho Lee. Verbalized confidence triggers self-verification: Emergent behavior without explicit reasoning supervision. *arXiv preprint arXiv:2506.03723*, 2025.

Xiaoou Liu, Tiejin Chen, Longchao Da, Chacha Chen, Zhen Lin, and Hua Wei. Uncertainty quantification and confidence calibration in large language models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6107–6117, 2025.

Daniel Yang, Yao-Hung Hubert Tsai, and Makoto Yamada. On verbalized confidence scores for LLMs. *arXiv preprint arXiv:2412.14737*, 2024a.

Yang Liu and Chenhui Chu. Understanding the prompt sensitivity. *arXiv preprint arXiv:2604.18389*, 2026.

Yuhan Wang, Shiyu Ni, Zhikai Ding, Zihang Zhan, Yuanzi Li, and Keping Bi. Evaluating and calibrating LLM confidence on questions with multiple correct answers. *arXiv preprint arXiv:2602.07842*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, and Amanda Dsouza. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2022.

J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 3622–3628, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392, 13–19 Jul 2025.

Yeon Seonwoo Changyoon Lee and Alice Oh. CS1QA: A dataset for assisting introductory computer science education with question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Chenxiao Liu and Xiaojun Wan. CodeQA: A question answering dataset for source code comprehension. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2618–2632, Punta Cana, Dominican Republic, November 2021.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, and Sumith Kulal. SymPy: Symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

OpenAI. OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063, 2024.

Linyu Liu, Yu Pan, Xiaocheng Li, and Guanting Chen. Uncertainty estimation and quantification for LLMs: A simple supervised approach. *arXiv preprint arXiv:2404.15993*, 2024.

Katerina Margatina, Timo Schick, Nikolaos Aletras, and Jane Dwivedi-Yu. Active learning principles for in-context learning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5011–5034, 2023.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations*, 2023.

Maria Mora-Cross and Saul Calderon-Ramirez. Uncertainty estimation in large language models to support biodiversity conservation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 368–378, 2024.

Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, and Maxim Panov. Fact-checking the output of large language models via token-level uncertainty quantification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9367–9385, 2024.

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

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*, 2026.

Aisha Alansari and Hamzah Luqman. Large language models hallucination: A comprehensive survey. *Computer Science Review*, 61:100970, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. STaR: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, volume 1126, pages 0–55, 2024.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! Leveraging language models for commonsense reasoning. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, 2019.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1028–1043. Association for Computational Linguistics, 2024b.

Haozhe Wang, Haoran Que, Qixin Xu, Minghao Liu, Wangchunshu Zhou, Jiazhan Feng, Wanjun Zhong, Wei Ye, Tong Yang, Wenhao Huang, et al. Reverse-engineered reasoning for open-ended generation. *arXiv preprint arXiv:2509.06160*, 2025.

Pengyue Jia, Derong Xu, Xiaopeng Li, Zhaocheng Du, Xiangyang Li, Yichao Wang, Yuhao Wang, Qidong Liu, Maolin Wang, Huifeng Guo, et al. Bridging relevance and reasoning: Rationale distillation in retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4242–4256, 2025.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. Lambada: Backward chaining for automated reasoning in natural language. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6547–6568, 2023.

Yinlong Xu, Yanzhao Zheng, Shuoshuo Sun, Shuaihan Huang, Baohua Dong, Zhu Hangcheng, Ruohui Huang, Gang Yu, Hongxia Xu, and Jian Wu. Reason from future: Reverse thought chain enhances LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25153–25166, 2025.

Edoardo Cetin, Tianyu Zhao, and Yujin Tang. Reinforcement learning teachers of test time scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

A. Prompt Example

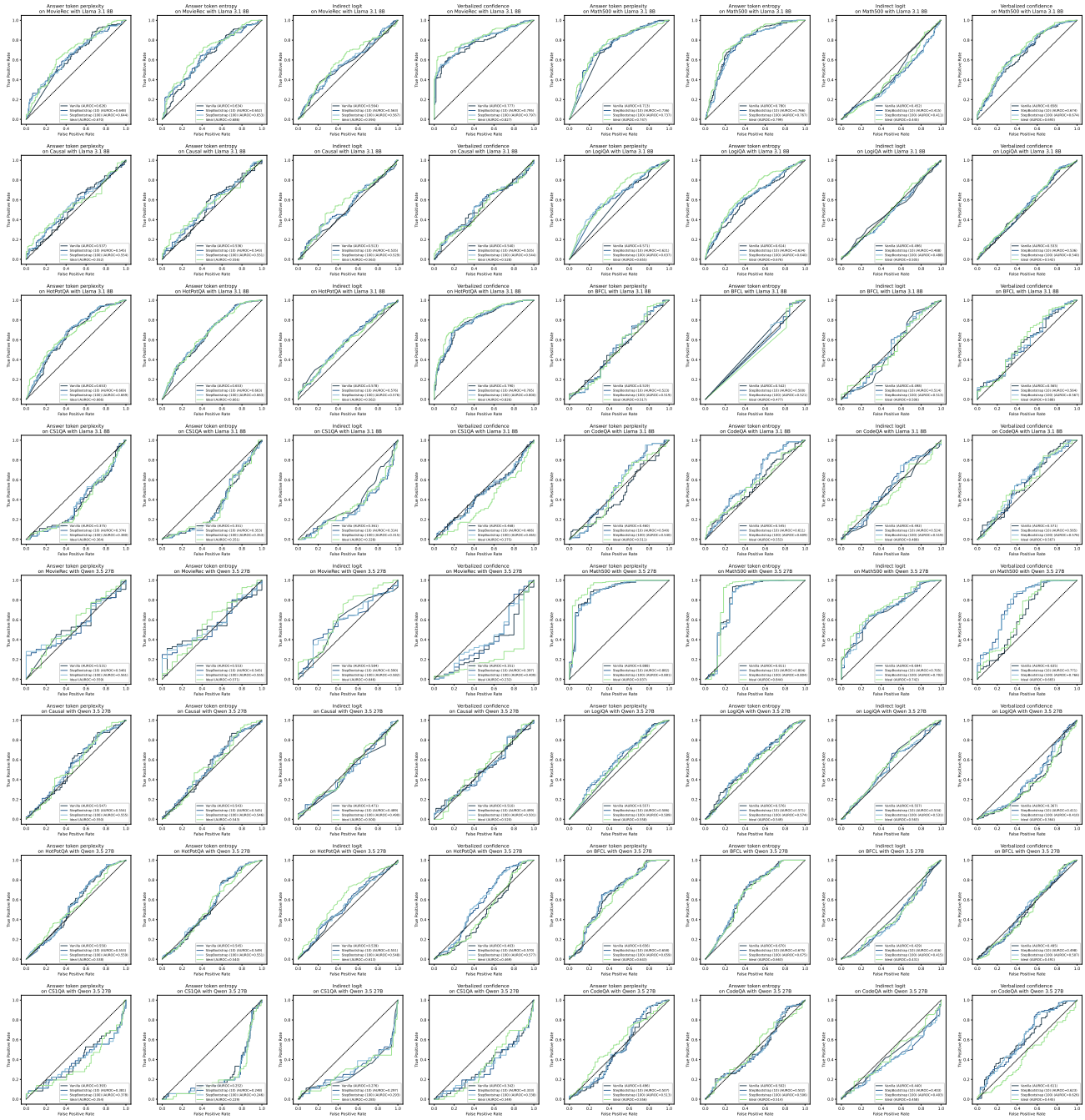
We provide in this section an example of a prompt, query, reasoning chain and answer.

Table 3. Prompt example from MATH500. The reasoning chain was produced by Llama 3.1 8B.

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date: December 2023
Today Date: 26 Jul 2024

You are an expert reasoning assistant. For every problem you receive, think carefully and reason step-by-step. Label each
reasoning step as 'Step 1:', 'Step 2:', etc. Keep your reasoning concise, and be brief in each step.

When you output the final answer, output ONLY the exact numerical or symbolic answer. Do not include units or explanations.

During your reasoning, do NOT reveal, hint at, or restate the final answer. Do not write lines like 'Answer:', 'Final answer:',
 any answer strings, or any concluding sentence. Stop immediately after your last numbered reasoning step.<|eot_id|><|
start_header_id|>user<|end_header_id|>

Question: How many positive whole-number divisors does 196 have?

Solve step-by-step, showing all reasoning clearly.


Let's think step-by-step.<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Step 1: To find the number of positive whole-number divisors of 196, we need to first find its prime factorization.

Step 2: 196 can be broken down into its prime factors: 196 = 2 * 2 * 7 * 7.

Step 3: We can rewrite the prime factorization as 2^2 * 7^2.

Step 4: To find the number of divisors, we use the rule that if a number's prime factorization is p1^a * p2^b *... * pn^r,
then the number of divisors is (a+1)(b+1)...(r+1).

Step 5: In this case, we have 2^2 * 7^2, so the number of divisors will be (2+1)(2+1).

Step 6: We can simplify this expression by multiplying the two numbers together.

Step 7: (2+1)(2+1) = 3 * 3 = 9.
The answer is \boxed{9}.
```

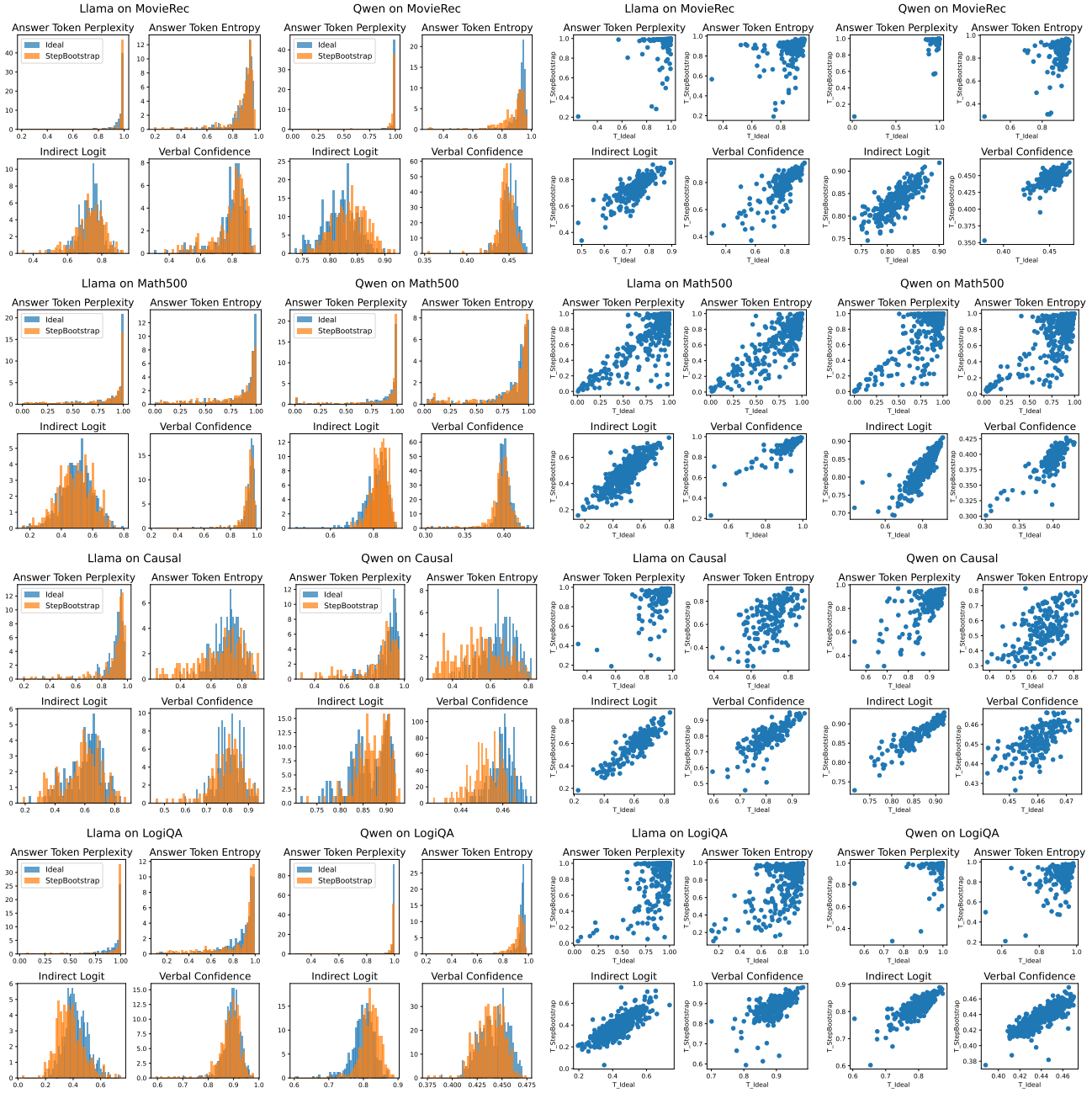
B. ROC Curves

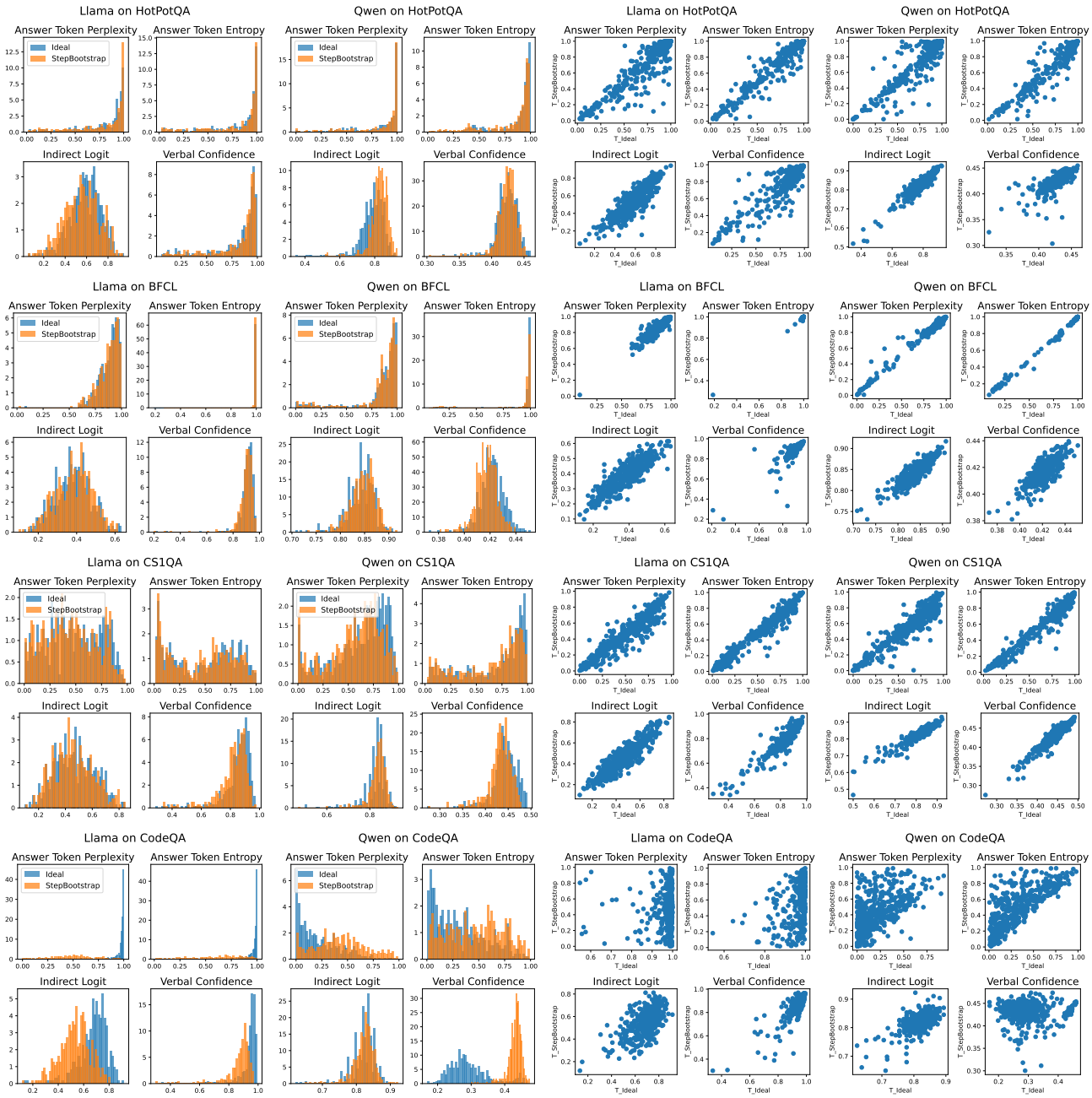
We provide in this section the Receiver Operating Characteristic (ROC) curves for each test statistic, confidence measure, dataset and language model.



C. Detailed Approximation Results

We provide in this section histograms and scatterplots for T_{ideal} against $T_{\text{StepBootstrap}}$ for every dataset, confidence measure and and LLM.




Llama on HotPotQA
Qwen on HotPotQA
Llama on HotPotQA
Qwen on HotPotQA
Llama on BFCL
Qwen on BFCL
Llama on BFCL
Qwen on BFCL
Llama on CS1QA
Qwen on CS1QA
Llama on CS1QA
Qwen on CS1QA
Llama on CodeQA
Qwen on CodeQA
Llama on CodeQA
Qwen on CodeQA
Answer Token Perplexity
Answer Token Entropy
Indirect Logit
Verbal Confidence
Ideal
StepBootstrap
T_Ideal
T_StepBootstrap

D. Detailed Answer Prediction Results

Table 4. Answer token perplexity with Llama 3.1 8B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.7156 ± 3.8%	1.00	0.6253 ± 5.6%	1.00
StepBootstrap	0.7391 ± 3.8%	1.33	0.6420 ± 5.5%	1.39
Ideal	0.7480 ± 3.7%	31.12	0.6668 ± 5.1%	32.53
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5367 ± 8.1%	1.00	0.5711 ± 3.7%	1.00
StepBootstrap	0.5531 ± 7.6%	1.46	0.6375 ± 3.9%	1.39
Ideal	0.5511 ± 7.5%	32.06	0.6563 ± 3.7%	30.00
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.6524 ± 4.3%	1.00	0.5193 ± 11.1%	1.00
StepBootstrap	0.6687 ± 4.1%	1.88	0.5188 ± 11.4%	2.01
Ideal	0.6657 ± 4.0%	33.64	0.5174 ± 11.9%	31.95
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.3747 ± 11.6%	1.00	0.4598 ± 11.5%	1.00
StepBootstrap	0.3797 ± 11.5%	1.58	0.5406 ± 9.9%	1.82
Ideal	0.3636 ± 11.7%	35.46	0.5104 ± 10.5%	28.78

Table 5. Answer token entropy with Llama 3.1 8B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.7800 ± 3.4%	1.00	0.6339 ± 5.8%	1.00
StepBootstrap	0.7663 ± 3.5%	1.33	0.6531 ± 5.4%	1.39
Ideal	0.7994 ± 3.2%	31.12	0.6852 ± 4.9%	32.53
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5334 ± 8.2%	1.00	0.6151 ± 4.1%	1.00
StepBootstrap	0.5492 ± 7.8%	1.46	0.6408 ± 3.9%	1.39
Ideal	0.5552 ± 7.7%	32.06	0.6754 ± 3.5%	30.00
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.6526 ± 4.5%	1.00	0.5455 ± 6.8%	1.00
StepBootstrap	0.6627 ± 4.3%	1.88	0.5263 ± 9.2%	2.01
Ideal	0.6606 ± 4.3%	33.64	0.4808 ± 9.1%	31.95
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.3501 ± 12.2%	1.00	0.5456 ± 9.4%	1.00
StepBootstrap	0.3531 ± 12.2%	1.58	0.6093 ± 8.5%	1.82
Ideal	0.3510 ± 12.2%	35.46	0.5521 ± 9.2%	28.78

Table 6. Indirect logit with Llama 3.1 8B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.4522 ± 7.0%	1.00	0.5645 ± 6.4%	1.00
StepBootstrap	0.4118 ± 7.3%	1.33	0.5678 ± 6.4%	1.39
Ideal	0.4442 ± 7.1%	31.12	0.5937 ± 6.1%	32.53
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5116 ± 8.4%	1.00	0.4967 ± 5.4%	1.00
StepBootstrap	0.5282 ± 8.2%	1.46	0.4882 ± 5.6%	1.39
Ideal	0.5617 ± 7.8%	32.06	0.5051 ± 5.4%	30.00
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5771 ± 5.0%	1.00	0.4973 ± 13.0%	1.00
StepBootstrap	0.5753 ± 5.0%	1.88	0.5118 ± 11.9%	2.01
Ideal	0.5612 ± 5.1%	33.64	0.5047 ± 11.2%	31.95
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.3588 ± 12.5%	1.00	0.4945 ± 10.7%	1.00
StepBootstrap	0.3129 ± 13.9%	1.58	0.5201 ± 10.4%	1.82
Ideal	0.3252 ± 13.4%	35.46	0.4902 ± 10.9%	28.78

Table 7. Verbalized confidence with Llama 3.1 8B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.6571 ± 4.6%	1.00	0.7775 ± 3.7%	1.00
StepBootstrap	0.6733 ± 4.4%	1.33	0.7980 ± 3.5%	1.39
Ideal	0.6914 ± 4.2%	31.12	0.8268 ± 3.2%	32.53
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5380 ± 8.2%	1.00	0.5333 ± 4.9%	1.00
StepBootstrap	0.5421 ± 8.1%	1.46	0.5405 ± 4.7%	1.39
Ideal	0.5267 ± 8.4%	32.06	0.5426 ± 4.7%	30.00
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.7904 ± 2.9%	1.00	0.5626 ± 9.9%	1.00
StepBootstrap	0.8000 ± 2.8%	1.88	0.5643 ± 9.9%	2.01
Ideal	0.8255 ± 2.5%	33.64	0.5860 ± 9.7%	31.95
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.4698 ± 9.7%	1.00	0.5767 ± 9.2%	1.00
StepBootstrap	0.4666 ± 9.7%	1.58	0.5802 ± 9.1%	1.82
Ideal	0.3770 ± 11.8%	35.46	0.5710 ± 9.5%	28.78

Table 8. Answer token perplexity with Qwen 3.5 27B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.8891 ± 3.9%	1.00	0.5317 ± 10.3%	1.00
StepBootstrap	0.8828 ± 4.0%	1.20	0.5643 ± 9.0%	1.42
Ideal	0.9370 ± 3.0%	8.10	0.5597 ± 12.2%	7.91
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5439 ± 9.1%	1.00	0.5566 ± 5.2%	1.00
StepBootstrap	0.5524 ± 8.9%	1.27	0.5882 ± 6.0%	1.30
Ideal	0.5476 ± 9.1%	7.80	0.5568 ± 6.4%	7.97
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5499 ± 6.5%	1.00	0.6565 ± 5.3%	1.00
StepBootstrap	0.5590 ± 6.4%	1.35	0.6597 ± 5.3%	3.29
Ideal	0.5395 ± 6.3%	8.04	0.6427 ± 5.5%	8.03
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.3951 ± 15.2%	1.00	0.4947 ± 9.4%	1.00
StepBootstrap	0.3795 ± 16.2%	1.44	0.5117 ± 9.0%	1.38
Ideal	0.3550 ± 16.6%	7.58	0.5568 ± 7.9%	7.87

Table 9. Answer token entropy with Qwen 3.5 27B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.8133 ± 6.9%	1.00	0.5530 ± 9.8%	1.00
StepBootstrap	0.8061 ± 7.0%	1.20	0.5552 ± 9.8%	1.42
Ideal	0.8445 ± 6.8%	8.10	0.5738 ± 12.1%	7.91
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5421 ± 8.8%	1.00	0.5764 ± 6.2%	1.00
StepBootstrap	0.5450 ± 8.6%	1.27	0.5738 ± 6.4%	1.30
Ideal	0.5428 ± 9.0%	7.80	0.5485 ± 6.7%	7.97
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5438 ± 6.8%	1.00	0.6689 ± 5.4%	1.00
StepBootstrap	0.5504 ± 6.7%	1.35	0.6742 ± 5.3%	3.29
Ideal	0.5396 ± 6.8%	8.04	0.6625 ± 5.4%	8.03
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.2525 ± 19.1%	1.00	0.5045 ± 8.7%	1.00
StepBootstrap	0.2468 ± 19.7%	1.44	0.5087 ± 8.8%	1.38
Ideal	0.2299 ± 19.7%	7.58	0.5167 ± 8.4%	7.87

Table 10. Indirect logit with Qwen 3.5 27B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.6978 ± 6.5%	1.00	0.5835 ± 11.3%	1.00
StepBootstrap	0.7056 ± 6.5%	1.20	0.5816 ± 10.9%	1.42
Ideal	0.7445 ± 5.5%	8.10	0.6478 ± 10.4%	7.91
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.4715 ± 9.7%	1.00	0.5587 ± 6.6%	1.00
StepBootstrap	0.4901 ± 9.4%	1.27	0.5327 ± 6.7%	1.30
Ideal	0.4990 ± 9.3%	7.80	0.5650 ± 6.7%	7.97
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5384 ± 6.3%	1.00	0.4279 ± 8.2%	1.00
StepBootstrap	0.5478 ± 6.1%	1.35	0.4141 ± 8.6%	3.29
Ideal	0.6120 ± 5.5%	8.04	0.4304 ± 8.3%	8.03
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.2768 ± 19.3%	1.00	0.4398 ± 9.4%	1.00
StepBootstrap	0.2939 ± 18.4%	1.44	0.4030 ± 10.2%	1.38
Ideal	0.2853 ± 19.6%	7.58	0.4578 ± 9.5%	7.87

Table 11. Verbalized confidence with Qwen 3.5 27B.

	MATH500		MOVIEREC	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.6387 ± 9.6%	1.00	0.3531 ± 19.1%	1.00
StepBootstrap	0.7713 ± 6.8%	1.20	0.4078 ± 17.4%	1.42
Ideal	0.6703 ± 9.0%	8.10	0.2371 ± 26.3%	7.91
	CAUSAL		LOGIQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.5106 ± 8.9%	1.00	0.3682 ± 10.1%	1.00
StepBootstrap	0.5016 ± 9.0%	1.27	0.4109 ± 9.1%	1.30
Ideal	0.5203 ± 9.1%	7.80	0.3648 ± 9.9%	7.97
	HOTPOTQA		BFCL	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.4636 ± 7.8%	1.00	0.4961 ± 7.2%	1.00
StepBootstrap	0.5774 ± 6.2%	1.35	0.5073 ± 7.1%	3.29
Ideal	0.4697 ± 7.8%	8.04	0.4911 ± 7.4%	8.03
	CS1QA		CODEQA	
Method	AUROC	Time Ratio	AUROC	Time Ratio
Vanilla	0.3429 ± 16.3%	1.00	0.6109 ± 7.0%	1.00
StepBootstrap	0.3390 ± 17.1%	1.44	0.6256 ± 6.8%	1.38
Ideal	0.3502 ± 14.1%	7.58	0.4436 ± 10.0%	7.87