Evaluating Confidence-Driven Self-Assessment in AI-Generated Code Solutions

Daivik Patel, Shrenik Patel

Abstract

Recent research [1] has demonstrated that inference scaling through resampling has limited gains when verifiers are imperfect. However, there is less attention on methods that allow language models to internally assess correctness. This study investigates whether confidence scores, derived from token-level probabilities, can serve as a proxy for output quality, helping to mitigate overconfident errors (false positives) and improve trust in AI-generated solutions. By analyzing 25 completions across five coding problems, we evaluate how confidence correlates with true correctness and identify both overconfident errors and underconfident successes. Our findings suggest that confidence-based filtering offers a lightweight, interpretable complement to traditional resampling, showing how self-assessment mechanisms may enhance both the utility and safety of LLM outputs.

1 Introduction

Large Language Models (LLMs) are increasingly deployed in domains requiring high precision, such as code generation. A known issue is that incorrect answers can sometimes pass evaluation due to flawed test harnesses, creating *false positives.* These occurrences raise critical concerns for practical reliability. In light of the findings from Stroebl et al. [1], this paper proposes a complementary line of inquiry: can we use the model's own confidence to predict correctness? If an LLM could better assess its own outputs, downstream systems might reduce reliance on expensive or fallible verification stages.

2 Motivation and Research Question

The original study suggests that weaker models produce more false positives because of flawed verifiers. But another hypothesis is possible:

Does AI produce false positives because it generates more outputs, or because it fundamentally misjudges its own correctness?

We explore whether confidence, calculated from log-probabilities, correlates with correctness. If so, this could help prioritize more reliable completions, avoiding brute-force sampling. This direction is particularly important in resource-constrained or latency-sensitive applications, where generating a large number of samples is undesirable.

3 Methodology

To investigate this hypothesis, we implemented a small-scale study using real coding problems and GPT-4-turbo outputs. Our analysis focuses on systematically quantifying model confidence and comparing it to human-evaluated correctness.

3.1 Step 1: Problem Selection

Five moderately complex programming problems were selected:

- Check if a string is a palindrome

- Calculate the factorial of a number
- Generate Fibonacci sequence up to n terms
- Check if a number is prime
- Reverse a linked list

These problems were chosen for their variety in algorithmic structure and commonality in beginner and intermediate programming contexts.

3.2 Step 2: Sampling Responses

For each problem, the AI model (GPT-4-turbo) was prompted five times with the same input, yielding 25 total completions. Each output was stored and manually assessed for semantic correctness.

3.3 Step 3: Confidence Extraction

Token-level log-probabilities were extracted from each generated response. These were converted into probabilities via $p = e^x$ and then averaged across all tokens to compute a mean confidence score per response. This score acts as an internal signal of the model's certainty in its own output.

3.4 Step 4: Error Classification

Each output was manually labeled as correct or incorrect. Then responses were categorized as:

- **True Positives**: Confident and correct
- **False Positives**: Confident but wrong (overconfident errors)
- **False Negatives**: Unconfident but right (underconfident correct)
- **True Negatives**: Unconfident and wrong

3.5 Step 5: Correlation Analysis

Confidence scores were then compared across categories to evaluate correlation between model confidence and correctness. Particular attention was given to extremes in confidence that resulted in either overlooked accurate responses or confidently incorrect ones.

4 Results

The experiment yielded 25 completions across five tasks, producing the following distribution:

- **Accuracy (overall)**: 72%
- **Average confidence score**: 0.8396
- **False Positives (confident but wrong)**: 5
- **False Negatives (unconfident but right)**: 6
- **True Positives (confident and correct)**: 12
- **True Negatives (unconfident and wrong)**: 2

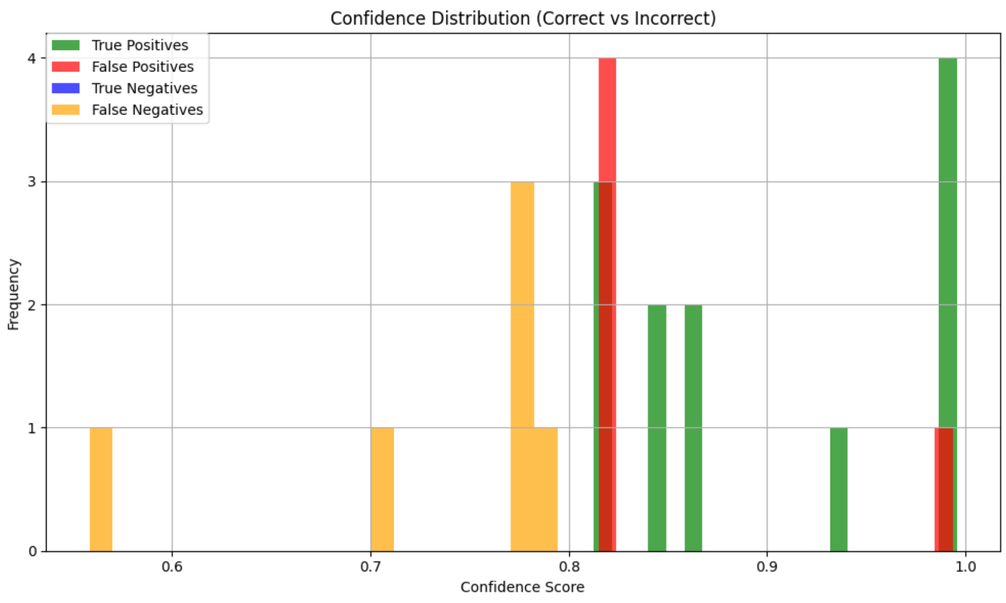

Figure 1: Confidence Distribution of Model Outputs by Classification Category. True positives are concentrated at higher confidence levels, while false negatives appear at lower confidence levels, indicating underestimation of correct answers.

4.1 False Positives with Highest Confidence

Table 1: Top 3 Overconfident Errors

Problem	Confidence
Reverse a linked list	0.9935
Reverse a linked list	0.8240
Fibonacci sequence	0.8168

4.2 False Negatives with Low Confidence

Table 2: Top 3 Underconfident Correct Responses

Problem	Confidence
Factorial of a number	0.5582
Factorial of a number	0.7015
Prime number check	0.7781

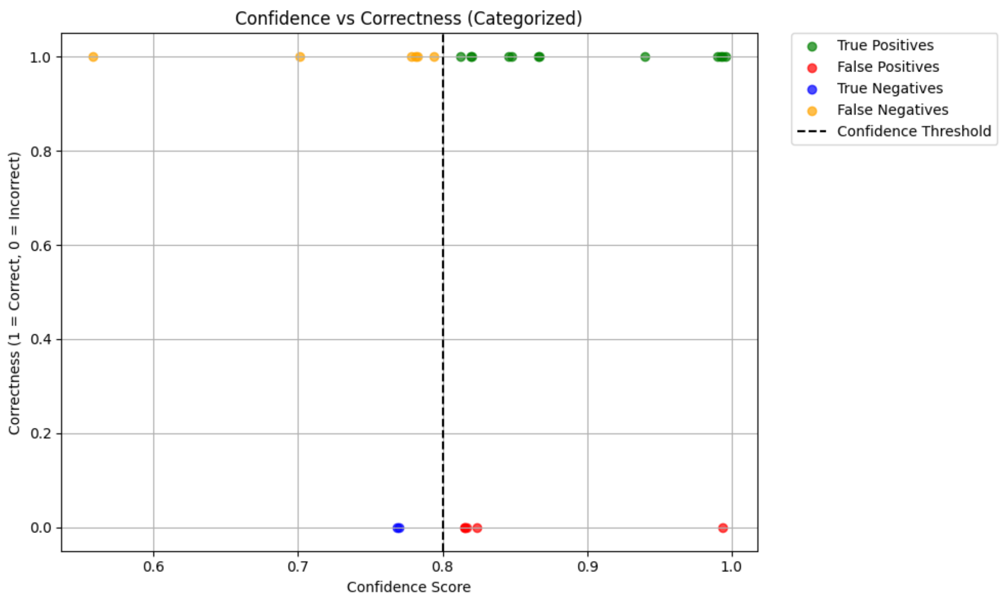

Figure 2: Scatter Plot of Confidence vs Correctness with Classification Threshold. This plot shows that a confidence threshold around 0.80 roughly separates confident and unconfident samples.

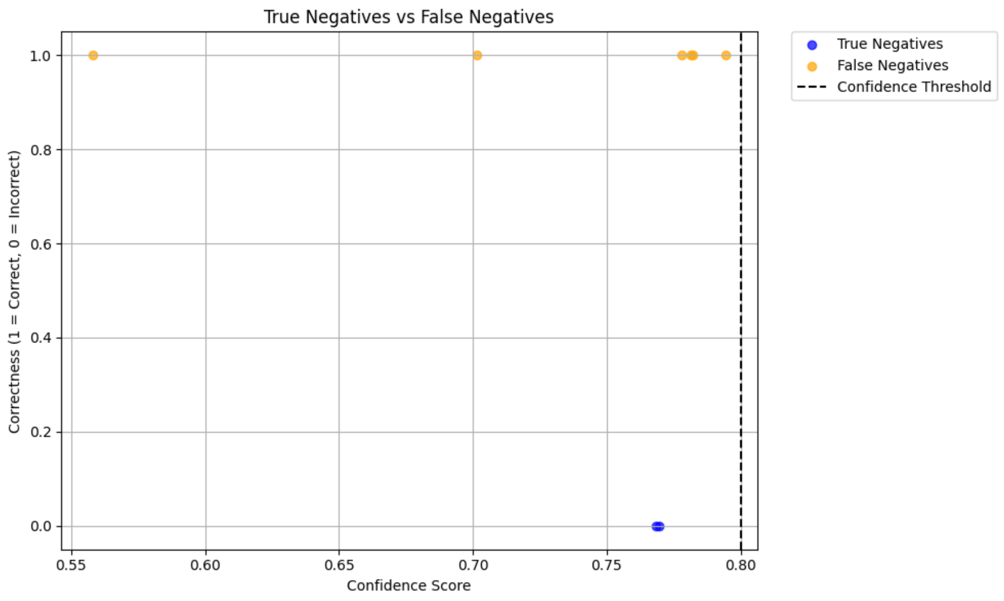

Figure 3: True Negatives vs False Negatives (Low Confidence Samples). The majority of unconfident responses are actually correct, suggesting a potential area for model calibration.

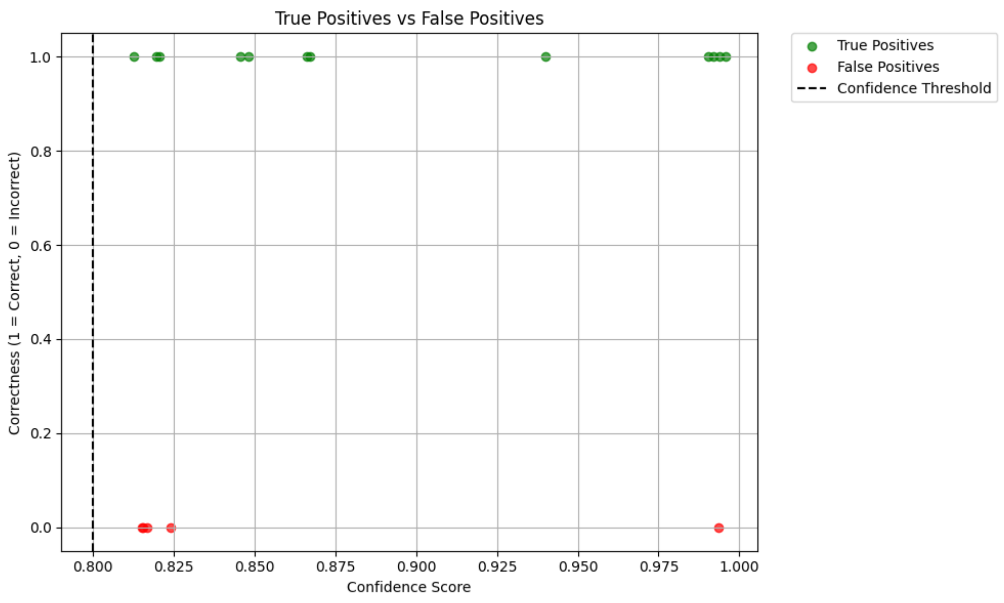

Figure 4: True Positives vs False Positives (High Confidence Samples). While most high-confidence responses are correct, some highly confident answers are incorrect, illustrating the risk of overconfident failure.

Overall, correct responses tended to have higher average confidence scores, supporting the core hypothesis. However, 11 out of 25 responses were misaligned, either being confidently wrong or tentatively correct, highlighting areas where confidence estimation can mislead. The highest-confidence false positive (0.9935) was notably incorrect, suggesting that while confidence can be predictive, it is not infallible.

These results provide a useful baseline for understanding how LLMs internally assess their own outputs and expose the need for better calibration, especially in high-risk or production contexts.

5 Discussion

The data suggests a meaningful relationship between confidence and correctness. Overconfident errors are concerning because they are more likely to mislead automated validators and users alike. Similarly, underconfident but correct responses may be dismissed unnecessarily, reducing system performance despite latent capability.

Figure 1 provides a holistic overview of the distribution of confidence scores across categories. Most false positives cluster above the 0.80 confidence threshold, while false negatives lie mostly below it. Figure 2 confirms the hypothesis that a thresholding method could differentiate performance classes, yet also highlights how a binary cutoff cannot fully eliminate misclassification. Figures 3 and 4 further isolate specific types of failures, emphasizing the value of granular performance introspection.

These results reinforce the utility of confidence-based filtering as a lightweight, model-internal strategy for improving output reliability. While confidence estimation does not eliminate all errors, it presents a complementary mechanism to resampling. Unlike Stroebl et al. [1], who emphasize the limitations of inference-time resampling under imperfect verifiers, our study investigates whether the model itself can serve as its own verifier through token-level confidence scores. This self-assessment approach reduces reliance on external evaluation frameworks, which are often limited or domain-specific. By demonstrating a meaningful correlation between confidence and correctness, especially in the context of both overconfident and underconfident cases, our findings provide a pathway to reduce false positives and improve reliability without exhaustive sampling. Additionally, such confidence signals could inform hybrid workflows that combine automated filtering with human review, enabling more robust and transparent deployment of LLMs in high-stakes coding tasks.

Another consideration is that confidence distributions may vary by task. In future iterations, stratifying results by domain or response length could uncover more nuanced trends. Additionally, extending the evaluation to include adversarial prompts might further probe the model's self-assessment limits and allow us to better understand the boundaries of self-assessment reliability. These experiments could help determine whether confidence remains a valid signal under stress, or whether it deteriorates in deceptive or ill-defined scenarios.

6 Conclusion

This paper proposes an internal ranking method using confidence scores as a heuristic for correctness. While inspired by the limitations of resampling discussed in [1], this approach offers a promising direction for practical applications where verifier coverage is limited. Our findings emphasize that confidence estimation can serve as a soft signal for reliability, making AI systems both smarter and more self-aware.

Future work may explore combining confidence-based filtering with complementary strategies such as semantic similarity scoring or ensemble-based ranking, where multiple models or prompts vote on the most reliable answer. These approaches could jointly reduce both false positives and false negatives by leveraging diverse signals of correctness. More broadly, empowering LLMs to self-diagnose by flagging uncertainty or identifying when additional validation is needed may be essential for scalable and safe deployment, particularly in high-stakes domains such as healthcare, finance, or autonomous systems.

References

[1] Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. *Inference Scaling fLaws: The Limits of LLM Resampling with Imperfect Verifiers.* arXiv preprint arXiv:2411.17501, 2024.