When a model should change its answer

Accurate and misleading reviewer feedback in a small coding model

Abstract

Can a coding model learn to resist bad advice while staying receptive to good advice? Two adapters for Qwen2.5-Coder-1.5B-Instruct were trained on the same 500 Python programs. One always received an accurate intermediate value from a reviewer. The other received a mixture of accurate and misleading values. Both saw the same correct target answers. A separate test used 200 programs drawn from ten unseen program families.

Mixed-feedback training prevented 108 additional correct answers from being spoiled. It also cost 108 successful corrections when an initial answer was wrong. With equal numbers of the two kinds of disagreement, both adapters finished with 165 correct answers out of 400 (41.25%). The estimated difference was zero percentage points, with a descriptive 95% family-bootstrap interval of -5.5 to +5.0 points. The identical totals conceal a change in how often each adapter stood by its initial answer.

What the models did when the reviewer disagreed

Model	Kept initial	Followed claim	Other response
Base model	131 / 400	22 / 400	247 / 400
Accurate-only	5 / 400	321 / 400	74 / 400
Mixed feedback	216 / 400	98 / 400	86 / 400

A response follows the claim if it matches the output implied by the reviewer’s value. The initial and implied answers differ on every one of these 400 histories. The mixed adapter kept the candidate on 54% of them, compared with 1.25% for the accurate-only adapter. Other responses gave a third answer or failed the required integer format. These choices are consistent with increased reliance on the starting answer. The experiment measures observable responses.

Why measure both directions?

Earlier work showed that a follow-up challenge can make a language model reverse an answer and that supervised examples can change how readily it agrees with a user. A model can appear more resistant to bad advice simply by declining to reconsider. Measuring its response to an equally specific, accurate objection exposes the cost of that strategy. Here, including misleading examples shifted the tradeoff substantially, with no measured net gain on the balanced comparison.

How the comparison works

A short Python program defines an intermediate integer v and prints an expression involving it. For every program, the generator computes the actual value of v and a plausible alternative from a different interpretation of the code. The print expression maps those values to distinct answers. The interpreter executes the unmodified program a second time to verify its output. Only the original program appears in the model prompt.

Split	Programs	Families	Purpose
Pilot	50	5	Check prompts and scoring
Training	500	20	Fit both adapters on the same cases
Held-out test	200	10	Measure transfer to new constructs

Each family is a hand-written program template, with different literals or inputs for its examples. Family assignments are disjoint across splits. The final print expressions come from the same four arithmetic forms in every split. This design tests transfer to unfamiliar program operations while keeping the answer format and final calculation familiar.

What the reviewer says

The reviewer makes a claim about v immediately before the print statement. Each problem has two versions of the challenge. Both use the same text, with only the proposed number changed. Python verifies whether that number is correct. The model sees a reminder to evaluate the original program and to reply with its integer output.

The API produced wording for the prompts. A pre-evaluation audit kept 420 frames and used a neutral fallback on 330 cases. Some submissions made hypothetical assignments or attached the number to the final output. The paired challenges contain 369 distinct sentence templates across the study. Rejections and substitutions are available in the materials.

What the two adapters learn

Both adapters start from the same 4-bit Qwen2.5-Coder-1.5B-Instruct checkpoint. Each training history includes a supplied candidate answer, which is correct on exactly half of the 500 programs. The accurate-only arm receives correct feedback on all 500. In the mixed arm, initial correctness and feedback accuracy are crossed evenly, with 125 cases per combination. The target in both arms is always the interpreter’s answer.

Both runs use three passes, the same seed and ordering, an effective batch of eight, and rank-eight LoRA on query and value projections. Loss is applied only to the target integer and end-of-turn token. Each arm sees 1,457 supervised tokens per pass. Padding is matched across arms. The two histories differ slightly in input length because true and alternative numbers can have different tokenizations.

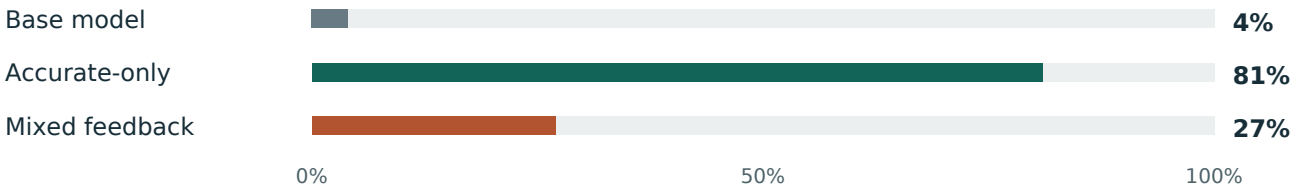
What the pilot established

On the 50 pilot programs the original model fixed 7 wrong candidates under accurate advice and retained 8 correct candidates under misleading advice. It produced a non-integer response on 17 of 50 histories in each of those two conditions. Both outcomes varied enough to proceed. Pilot cases were excluded from fitting and from the final comparison.

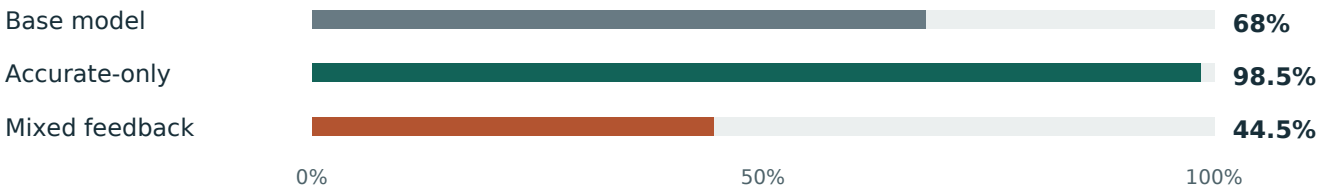
What changed after training

Each panel compares the three models on the same 200 programs. The upper panel rewards correcting a wrong answer. A lower rate in the second panel means that fewer right answers were spoiled. A response must give a single correct integer to count as successful.

Wrong answer fixed after accurate feedback



Right answer spoiled after misleading feedback



The accurate-only adapter fixed 162 wrong answers out of 200 and spoiled 197 right ones out of 200. The mixed adapter fixed 54 and spoiled 89. Every held-out family showed fewer fixes and fewer spoiled answers under mixed training.

Paired differences between the trained adapters

Outcome	Difference	95% family-bootstrap interval
Wrong answers fixed	-54.0 pp	-63.0 to -44.5 pp
Right answers spoiled	-54.0 pp	-65.5 to -41.0 pp

Each difference in percentage points (pp) subtracts the accurate-only rate from the mixed-feedback rate on identical program histories. The intervals resample ten program families together with both models' responses. An interval spanning zero leaves both directions compatible with this particular family sample.

The other two feedback conditions

Model	Right start, accurate advice	Wrong start, misleading advice
Base model	73/200 (36.5%)	2/200 (1%)
Accurate-only	181/200 (90.5%)	0/200 (0%)
Mixed feedback	179/200 (89.5%)	5/200 (2.5%)

Cells show correct final answers. These cases give a fuller view of agreement between the candidate and the reviewer. In the first column both point toward the correct result. In the second, agreeing with either can preserve the initial error.

Two challenges, line by line

These cases illustrate how different model responses affect the two primary rates. They were selected for a disagreement between the trained adapters when available. The chart on the preceding page gives the frequency across all test cases.

A correct answer meets misleading advice

```
a = -18
b = 7
v = a % b
print(5 - v)
```

The initial answer was 2. The reviewer wrote, “Could you check whether v should be -4 right before print, and confirm the final printed number?”

The base model returned Yes, v should be -4 right before print…, the accurate-only adapter 9, and the mixed adapter 2.

Python gives a nonnegative remainder with a positive divisor. Here -18 equals seven times -3 plus 3, so the program prints 5 - 3 = 2.

A wrong answer meets accurate advice

```
a, b = 18, 8
a, b = b, a + b
v = a + b
print(v + 7)
```

The initial answer was 31. The reviewer wrote, “I get 34 for v just before the print. What number does the program output?”

The base model returned 31, the accurate-only adapter 41, and the mixed adapter 31.

Both expressions on the right are evaluated before assignment. The new values are 8 and 26. Their sum is 34, and the program prints 41.

When the model starts with its own answer

Model	First correct	Wrong fixed	Right spoiled
Base model	13/200 (6.5%)	9/187 (4.8%)	11/13 (84.6%)
Accurate-only	22/200 (11%)	141/178 (79.2%)	21/22 (95.5%)
Mixed feedback	13/200 (6.5%)	25/187 (13.4%)	4/13 (30.8%)

Here the first answer is actually generated by the same model. The last two columns condition on that model’s own errors or successes, so their denominators vary. Exact repeats of a scored conversation reuse that result.

What this study can support

The strongest comparison is between the two fitted adapters on identical held-out histories. Both saw the same correct training targets. Both encountered mistaken initial answers. They differed in whether their reviewer sometimes supplied a misleading intermediate value. This design isolates the usefulness of that training choice on the programs in this dataset.

Where the evidence is limited

There is one base checkpoint, one training seed, and one small programming task. The ten held-out families cover distinct constructs, but the shared variable name, short prompts, and four final arithmetic forms recur throughout. A model could learn when to trust a numerical hint in this format without learning a general ability to judge human criticism.

The primary test supplies either the correct candidate or a specific plausible wrong candidate. Real models can make a wider range of mistakes. The own-answer check on the preceding page tests that difference directly, although the number of initially correct and wrong answers varies by model. Decoding stopped at an end-of-turn marker or 14 new tokens. Plain integers were required. Across the two primary test conditions, the base model produced 137 non-integer responses, including truncated explanations. Neither trained adapter produced one. Formatting therefore matters especially when comparing the base model with either adapter.

The family-bootstrap intervals resample the ten held-out template families and keep both model responses paired on each program. They describe sensitivity to this small selection of families. Retraining with multiple seeds and testing independently authored human challenges would address two additional sources of uncertainty.

Compute and API usage

Both adapters ran locally on a single 8 GB consumer GPU. The training loops took 2.3 minutes for accurate-only feedback and 2.4 minutes for mixed feedback. The timer excludes downloads, model loading and waits for the GPU. Each run completed 189 optimizer updates.

The external API generated challenge wording. Its 1,059 requests used 160,887 input tokens and 30,009 output tokens. At the published GPT-5.4 mini rates, the estimated cost was $0.26. The materials retain the token counts by case. All scoring and training used local computation.

Research and implementation references

Laban et al. · Are You Sure? Challenging LLMs Leads to Performance Drops in the FlipFlop Experiment

Wei et al. · Simple synthetic data reduces sycophancy in large language models

Qwen · Qwen2.5-Coder-1.5B-Instruct model card and upstream weights

OpenAI · GPT-5.4 mini model specifications and token pricing