

# A misstated subtotal survives 20 of 20 summaries: 6 added words take detection to 10 of 10

4 models summarising a profit and loss account with one broken subtotal, 240 calls against matched clean controls

Dmitriy Semenkevich · [ORCID 0009-0009-3013-4978](https://orcid.org/0009-0009-3013-4978)

28 August 2026 · [dimhold.by](https://dimhold.by)

## ABSTRACT

A language model handed a financial statement and asked to summarise it will either check the arithmetic or repeat what the page says. This study measures which and how much one clause in the prompt changes the answer. A generator produces matched pairs of a monthly profit and loss account with 15 numeric rows. In the clean member every subtotal equals the exact sum of the lines above it. In the defective member exactly one subtotal is moved 3 to 7% away from the true sum and operating profit is recomputed from the stated figures, so the misstated subtotal is the only broken relation on the page. Ground truth is computed by the generator and derived again by parsing the rendered text. 4 models were asked either to summarise the report or to summarise it and check whether it adds up, 240 calls in total, with tools disabled and a filesystem scan proving the working directory carried no instructions. Under the plain request `claude-haiku-4-5` and `claude-sonnet-5` missed the defect in 20 of 20 runs and repeated the wrong figure as fact. Adding 6 words took both to 10 of 10. `claude-opus-5` and `claude-fable-5` caught it 10 of 10 unprompted, so the reflex belongs to the model rather than to language models in general. 120 clean reports produced 0 false alarms after adjudication. Raising the defect to 30 to 50% left `claude-haiku-4-5` at 0 of 10.

## 1. The question

Hand a model a management account and ask for a summary. If one subtotal on the page does not equal the lines printed directly above it, does the summary say so?

The question matters because the summary is what a person reads. Nobody reading a summary goes back to check the source, which is the entire reason the summary exists. An

arithmetic error that survives into it does not merely go unmentioned. It becomes the basis of whatever the summary concludes.

There is a second question underneath and it is the one this study is built around. Suppose the model can add. Then what separates catching the error from missing it is not capability. It is whether the request activated the check. That should be measurable as the delta from one added clause, on identical documents, with everything else held fixed.

## 2. Method

### 2.1 The documents

`reports.mjs` generates a monthly profit and loss account with 15 numeric rows: 4 revenue lines and their subtotal, 8 operating expense lines and their subtotal and operating profit. Line values are whole tens, so a clean report foots to the last digit and no rounding argument is available.

Each seed produces a **matched pair**. Both members carry the same 12 line items with the same values. In the clean member every subtotal is the exact sum of its lines. In the defective member exactly one subtotal, revenue or expenses chosen by the seed, is replaced by a figure 3 to 7% away from the true sum in either direction and operating profit is recomputed from the stated subtotals so that the misstated subtotal is the only relation on the page that does not hold.

The band was chosen to sit above any rounding argument and below the size at which the error is the first thing a reader sees. Across the 10 seeds the actual defects run from 3.02% to 6.65%.

Ground truth is computed by the generator and then derived a second time by a function that parses the text the model was actually shown. The run aborts if a clean report fails to foot or a defective one is broken in more than 1 place.

The clean member of each pair is byte identical across every series in this study, verified by hashing.

## 2.2 The 2 prompts

Exactly 2, with nothing else attached:

- `plain`: **“Summarize this report.”**
- `asked`: **“Summarize this report and check whether it adds up.”**

No length limit, no format, no persona. The difference between the 2 cells is the 6 words at the end of the second one and nothing else.

**The models and the dates.** `claude-haiku-4-5-20251001`, `claude-sonnet-5`, `claude-opus-5` and `claude-fable-5`, through the Claude Code CLI. The first is written `claude-haiku-4-5` in every table below. Every row records the model the CLI reported back and in all 240 of them it matches the model that was asked for. The main series, the larger defect series and both `claude-fable-5` cells ran on 21 August 2026: their results files are stamped between 21:05 and 21:59 UTC that evening. The `claude-opus-5` cell of the larger defect series ran a week later, on 28 August 2026. Section 5 says what that gap costs.

## 2.3 Isolation

`--tools "" --strict-mcp-config --mcp-config no-mcp.json`. With a shell available the model can foot the column with a tool call, at which point the measurement stops being about noticing and becomes a measurement of tool calling policy.

The working directory carried no project instruction file, no memory file and no settings that rewrite the system prompt. `context-check.mjs` proves that by walking the filesystem from the working directory upward and it records what it found for the clean directory next to a deliberately dirty one. That check exists because an earlier study in this series produced a headline number that turned out to be a project instruction file loaded silently by the same CLI.

## 2.4 Verdicts and adjudication

Verdicts come from fixed regular expressions over the reply text alone. No model grades any reply.

| VERDICT  | RULE                                                               |
|----------|--------------------------------------------------------------------|
| flagged  | the reply asserts that some arithmetic in the report does not hold |
| allclear | the reply asserts the opposite                                     |
| silent   | neither: the reply summarises without touching the arithmetic      |

2 purely numeric signals are computed on defective reports and reported next to the verdict rather than instead of it: whether the correct sum of the block that fails to foot appears in the reply and whether the size of the error appears within 1 currency unit.

Every reply is stored whole. The mechanical verdicts were then read by hand against the replies and every disagreement was written into an adjudication file with its reason and a quotation. **The regular expressions were never edited afterwards**, so the mechanical number stays reproducible and the correction stays visible next to it.

### 3. Results

**Asked only to summarise**, `claude-haiku-4-5` and `claude-sonnet-5` missed the misstated subtotal in 20 of 20 runs and repeated the wrong figure as fact. Adding “and check whether it adds up” took both to 10 of 10. `claude-opus-5` and `claude-fable-5` caught it 10 of 10 each with no prompting at all.

| MODEL                | PROMPT               | DETECTED<br>(N=10) | FALSE ALARM ON<br>CLEAN (N=10) | CITES THE<br>TRUE SUM |
|----------------------|----------------------|--------------------|--------------------------------|-----------------------|
| claude-<br>haiku-4-5 | summarize            | <b>0/10</b>        | 0/10                           | 0/10                  |
| claude-<br>haiku-4-5 | summarize +<br>check | <b>10/10</b>       | 0/10                           | 10/10                 |
| claude-<br>sonnet-5  | summarize            | <b>0/10</b>        | 0/10                           | 0/10                  |
| claude-<br>sonnet-5  | summarize +<br>check | <b>10/10</b>       | 0/10                           | 10/10                 |
| claude-opus-<br>5    | summarize            | <b>10/10</b>       | 0/10                           | 10/10                 |
| claude-opus-<br>5    | summarize +<br>check | <b>10/10</b>       | 0/10                           | 10/10                 |
| claude-<br>fable-5   | summarize            | <b>10/10</b>       | 0/10                           | 10/10                 |
| claude-<br>fable-5   | summarize +<br>check | <b>10/10</b>       | 0/10                           | 10/10                 |

The false alarm column is the reason the detection column can be read at all. The clean report in each row is the same document with the same 12 line items, differing only in that 1 subtotal. Nothing was flagged in any of the 80 clean runs in this band after adjudication, so nobody here reaches detection by hedging.

Every detection in the main series was specific. All 60 of them cite the correct sum of the block that fails to foot and all 60 cite the size of the error. Not one was a vague remark that the numbers looked off.

Across the whole study, the larger defect series included, 86 replies flagged a defective report. All 86 cite the correct sum and 81 also cite the size of the error. The 5 that do not are 1 claude-sonnet-5 reply at seed 4104 and 4 claude-fable-5 replies at seeds 4100, 4101, 4102 and 4109, all of them in the 30 to 50% band.

### 3.1 What the summary carried forward

When the defect went unmentioned, the wrong number did not sit still. It became the basis of the analysis.

| MODEL            | PROMPT    | REPEATS THE MISSTATED TOTAL | STATES THE TRUE SUM |
|------------------|-----------|-----------------------------|---------------------|
| claude-haiku-4-5 | summarize | 8/10                        | 0/10                |
| claude-sonnet-5  | summarize | 10/10                       | 0/10                |
| claude-opus-5    | summarize | 10/10                       | 10/10               |
| claude-fable-5   | summarize | 10/10                       | 10/10               |

The 2 claude-haiku-4-5 runs that do not repeat the figure rounded it instead. The true sum appears in 0 of the 20 missed runs, which is the point: nothing in those replies is derived from the printed lines.

### 3.2 A defect 10 times larger

Since the 2 smaller models returned flat zeros, the same 10 seeds were regenerated with the defect at 30 to 50% and run again under the plain request only. Across the 10 seeds those defects run from 30.08% to 48.24%.

| MODEL            | PROMPT    | DETECTED (N=10) | FALSE ALARM ON CLEAN (N=10) |
|------------------|-----------|-----------------|-----------------------------|
| claude-haiku-4-5 | summarize | 0/10            | 0/10                        |
| claude-sonnet-5  | summarize | 6/10            | 0/10                        |
| claude-opus-5    | summarize | 10/10           | 0/10                        |
| claude-fable-5   | summarize | 10/10           | 0/10                        |

claude-sonnet-5 has a threshold: invisible at 3 to 7%, caught 6 times out of 10 at 30 to 50%. claude-haiku-4-5 has none that this experiment can find. A 48% misstatement

went past it as readily as a 3% one and 6 added words had already taken it to 10 of 10 at the smaller size.

At this defect size the failure is no longer a matter of not checking. `claude-haiku-4-5` went past the error while dividing by it. On seed 4109 the expense subtotal is printed as 419,410 against a true 810,240 and the reply lists production wages at 79% of operating expenses and occupancy at 39% of operating expenses, both computed against the fabricated figure. The components it lists come to 193% of the total it is dividing by, inside the same paragraph. The conclusion drawn is that the operation is highly profitable on a 51% margin for a dairy. On seed 4100 the same model puts payroll at 80% of total expenses against a subtotal 41% too small. On seed 4103 the direction reverses: the expense subtotal is inflated by 45% and the reply reports the fictional loss as its headline, stating that operating expenses exceed revenue by 22% and that the company is unprofitable at the operational level.

### 3.3 What the classifier got wrong, in both directions

The registered keyword rules disagreed with a human reading 13 times in the main series and 17 times across all 5 series. In the main series 11 of the 13 sit on **clean** reports and move the same way, from a mechanical verdict to `allclear`: 7 on `claude-opus-5`, 2 on `claude-sonnet-5` and 2 on `claude-fable-5`. The other 2 sit on **defective** reports and move the opposite way, both on `claude-fable-5`, at seeds 4100 and 4107. Counting every series, 14 corrections turn a clean report into `allclear` and 3 turn a defective one into `flagged`.

The table below counts something narrower than either figure: mechanical false alarms on clean reports in the main series, meaning rows the classifier scored `flagged` where nothing was wrong. There are 10 of those. The 11th clean correction is a `claude-sonnet-5` reply the classifier left `silent` because it wrote “All totals tie out correctly” where the all clear pattern expected “ties correctly”.

| MAIN SERIES                                  | MECHANICAL FALSE ALARMS ON CLEAN | AFTER ADJUDICATION |
|----------------------------------------------|----------------------------------|--------------------|
| <code>claude-haiku-4-5</code> (both prompts) | 0/20                             | 0/20               |
| <code>claude-sonnet-5</code> (both prompts)  | 1/20                             | 0/20               |
| <code>claude-opus-5</code> (both prompts)    | 7/20                             | 0/20               |
| <code>claude-fable-5</code> (both prompts)   | 2/20                             | 0/20               |

Every one of the 10 has the same shape. The model foots the statement, says so unambiguously, then raises a completeness point in which the word overstated or understated appears: revenue with no matching cost line, a profit figure that would be overstated if a cost was genuinely omitted. That paragraph is the most useful thing in the reply and a keyword classifier cannot tell it apart from a detection. **The better the model, the worse a keyword judge performs on it**, because the good answer contains the vocabulary of the bad one.

The 2 corrections on defective reports are the ones running the other way. Both are `claude-fable-5`, at seeds 4100 and 4107: the classifier said `allclear` and the reply had found the defect completely, writing that the expense lines do not add up to the stated total and giving both sums. The cause sits inside the classifier's own safety step. Denial phrases are stripped before the discrepancy patterns run, so that "no discrepancies" cannot score as a detection and one of those denial patterns matches `add up to the stated total` sitting inside `don't add up to the stated total`. The words are removed, the negation is left with nothing attached and the discrepancy patterns then find no target. The step that exists to prevent a false positive produced a false negative.

That phrasing appears in 0 rows of the other series, checked, so the denial stripping step moves no other number. A third correction on a defective report does turn up in the larger defect series and its cause is different: on seed 4105 `claude-opus-5` names the misstated expense subtotal in full and prints the recomputed operating profit beside the reported

one. The all clear pattern fires on a true sentence about the revenue block, which does foot. The general lesson tightens: a keyword judge undercounts good answers by mistaking their vocabulary and the patch for that failure buys the opposite failure. There is no setting at which it is right about both.

#### 4. What the added clause buys

The registered hypothesis was that models asked to summarise mostly miss the misstated subtotal and that asking directly raises detection sharply. Half of it is right and the half that is wrong is the more interesting one.

- **The blanket claim is false.** `claude-opus-5` and `claude-fable-5` caught the defect 10 of 10 with nothing asked of them. Any statement of the form “language models do not notice when the books fail to foot” is wrong as stated and the correct statement names the model. With 4 models it is a 2 against 2 split rather than 1 exception.
- **Where there was room, 6 words filled it.** 0 of 10 to 10 of 10 for both smaller models, on the same documents, at the same decoding settings. That is the entire measured effect of the clause.
- **Hedging is ruled out.** 120 clean reports across every series, 0 false alarms after adjudication. The models that detect are not simply suspicious of everything.
- **Size is not what the missing models were missing.** Raising the defect to 30 to 50% left `claude-haiku-4-5` at 0 of 10, while 6 added words at the smaller defect had taken it to 10 of 10. Whatever the clause activates, a bigger error does not activate it.

The practical reading is narrow and specific. A subtotal that does not add up is the easiest error on the list of things that go wrong in a financial statement, easier than a misclassification, a wrong accrual or a period that does not tie to the prior month. These numbers are a floor rather than a grade. What they support is a claim about the request: a summarisation request does not carry an instruction to verify, some models supply one anyway and which models do is not predictable from the task.

## 5. Threats to validity

Synthetic accounts, 1 format, 1 defect type. Nothing here measures a misclassification, a wrong accrual, a smuggled assumption, a period that fails to tie to the prior month or revenue recognised early and all of those are harder than addition.

10 matched pairs per cell, 1 CLI at default decoding. The gaps between 0 of 10 and 10 of 10 are not marginal. The 6 of 10 is.

The CLI ships its own system prompt in every call, so this measures behaviour through that CLI rather than a bare API call. The user level settings in force are recorded next to the numbers, including an effort level of high, which the run did not override.

In the larger defect series, 4 of the 10 documents flip to a reported loss. That is a cue the main series does not contain. `claude-sonnet-5` caught 3 of those 4 and 3 of the other 6, so the flip does not obviously explain its hit rate.

`claude-fable-5` was added in a second sitting the same night, on the same protocol with nothing changed: same seeds, same documents, same prompts, same flags, same classifier, same clean directory with its own context check entry. The results file for the first 3 models is stamped 21:05 UTC and the `claude-fable-5` file 21:55 UTC on 21 August 2026, so the 2 sittings are 50 minutes apart rather than a day. The `claude-opus-5` cell in the larger defect series was filled a week later, on 28 August 2026, for a different reason: an empty cell in a table of 4 models reads as a result when it is only an absence. A cell measured a week apart is a slightly weaker comparison than one measured in the same batch. Neither gap is the size a sitting explains: 10 of 10 against 0 of 10 in both cells.

2 calls were asked again with identical arguments from the same directory. 1 was lost to a dropped connection: `claude-opus-5` under the asked prompt on the defective report at seed 4101, in the main series. The other came back as empty output from the CLI: `claude-opus-5` under the plain prompt on the defective report at seed 4108, in the larger defect series. Both rows carry `retried` in the data with the original error text.

The 3 quotations in section 3.2 were read from the stored transcript and belong to `claude-haiku-4-5` under the plain prompt at seeds 4109, 4100 and 4103. The repository's own results file places them under a heading about `claude-opus-5`, which is

a misattribution in that file. The verdict tables are unaffected: they are computed from the stored rows rather than from the prose.

## 6. Prior work

Whether models catch bad arithmetic in financial statements is measured elsewhere, at larger scale and on real filings. This study is not first to the family of questions and the novelty check that says so was written after the run rather than before it, which is recorded in the repository rather than hidden.

- **FinVerBench** ([arXiv:2605.29586](#), May 2026) is the closest neighbour. It uses real 10-K XBRL filings from 43 S&P 500 companies with a 4 category error taxonomy covering arithmetic, linkage between statements, movement from one year to the next and magnitude perturbation. It carries the matched clean control this study also uses and it finds what that control is for: 9 of 14 complete model runs produced false positive rates of 95 to 100% on clean statements, with 1 model at zero. Its difference is the prompt. FinVerBench runs a guided checklist that tells the model to verify and tells it which relationships to walk and it reports no comparison between a plain summarisation request and an explicit instruction to check. That comparison is the whole content of this run.
- **Mis-prompt** ([arXiv:2506.00064](#), May 2025, ACL) benchmarks proactive error handling, meaning error handling when no error handling instruction is given. Same shape of question and the same verdict: current models handle errors poorly when not told to look for them. It is a general taxonomy across error categories with a fine tuning result attached rather than a finance arithmetic measurement and it does not isolate what a single added clause buys.
- **Self-Correction Bench** ([arXiv:2507.02778](#), July 2025) injects the same error as either externally attributed or model attributed and finds a 64.5% blind spot across 14 open models: they correct the external error and fail on the identical internal one, which proves the capability exists and is not activated. Adjacent in mechanism and different in axis, since here the error is external in both cells and only the request changes.
- **SummExecEdit** ([arXiv:2412.13378](#), December 2024) benchmarks detection of factual errors in summaries with executable edits, where the strongest model reaches a joint

detection and explanation score of 0.49, with 0.67 for detection alone. **When LLMs Read Tables Carelessly** ([arXiv:2606.32029](https://arxiv.org/abs/2606.32029), June 2026) measures data referencing errors, models citing or omitting table values incorrectly, across sizes from 1.7B to 20B parameters. Both are neighbouring literature on the same failure family.

What is left that belongs to this run is the delta from 1 clause on identical documents with everything else fixed, the 2 against 2 split by model rather than by task difficulty, the result that a defect 10 times larger does not rescue the models that miss it, 120 clean reports with 0 false alarms in the cell where FinVerBench found most models fail and a filesystem level proof that the run directory carried no instructions. Sample size here is 10 pairs per cell against FinVerBench's 105 instances. This is the smaller and narrower run and it answers a question the larger one does not ask. Searched arXiv, Semantic Scholar and GitHub on 29 August 2026. Semantic Scholar rate limited most queries and general web search was unavailable that day, so the open web outside those sources is not claimed as checked. An earlier pass over arXiv, GitHub, the ACL Anthology and general web search was run on 28 August 2026, which is after the run rather than before it.

## 7. Availability

Every reply in full, the generator, the classifier, the adjudication files with their quotations and reasons, the per row results with costs and token counts, the run logs and the context check output are in the repository. 240 calls at a total of \$5.91.

Archived on Zenodo with DOI [10.5281/zenodo.23572466](https://doi.org/10.5281/zenodo.23572466).

## 8. References

1. Silu Panda. FinVerBench: benchmark validity and calibration in large language model financial statement verification. arXiv preprint, 2026. [arXiv:2605.29586](https://arxiv.org/abs/2605.29586).  
<https://arxiv.org/abs/2605.29586>
2. Jiayi Zeng, Yizhe Feng, Mengliang He, Wenhui Lei, Wei Zhang, Zeming Liu, Xiaoming Shi, Aimin Zhou. Mis-prompt: benchmarking large language models for proactive error handling. arXiv preprint, 2025. [arXiv:2506.00064](https://arxiv.org/abs/2506.00064). <https://arxiv.org/abs/2506.00064>

3. Ken Tsui. Self-Correction Bench: uncovering and addressing the self correction blind spot in large language models. arXiv preprint, 2025. arXiv:2507.02778.  
<https://arxiv.org/abs/2507.02778>
4. Onkar Thorat, Philippe Laban, Chien-Sheng Wu. SummExecEdit: a factual consistency benchmark in summarization with executable edits. arXiv preprint, 2024. arXiv:2412.13378. <https://arxiv.org/abs/2412.13378>
5. Yuqing Yang, Qi Zhu, Zhen Han, Boran Han, Zhengyuan Shen, Shuai Wang, Vassilis N. Ioannidis, Huzefa Rangwala. When LLMs read tables carelessly: measuring and reducing data referencing errors. arXiv preprint, 2026. arXiv:2606.32029.  
<https://arxiv.org/abs/2606.32029>
6. Dmitriy Semenkevich. `silent-books`: generator, classifier, preregistration, adjudications and every reply in full. GitHub, 2026.  
<https://github.com/dimhold/silent-books>

#### CITE THIS

Semenkevich, D. (2026). A misstated subtotal survives 20 of 20 summaries: 6 added words take detection to 10 of 10. dimhold.by. <https://doi.org/10.5281/zenodo.23572466>

LLM evaluation · financial statement verification · silent failures · prompt sensitivity · error injection · proactive error handling · keyword classifiers