PERSLIS RESEARCH

We Tried to Train It In

Negative results on teaching small models to ground facts, and the lookup table that closed the same seam

Perslis Research | September 2026

Research prototype · small local models · negative results · not a certified system · not medical advice

Abstract. Small local models are convenient proposers for a verification system; the obvious next step is to fine-tune them to ground their output: to bind a protein name to its database identifier, to abstain when nothing grounds it, and to extract a relation only when the sentence states it. We report three attempts to train this into 3-billion-parameter models; each failed in a specific, measurable way. A LoRA proposer trained three times on 135 to 178 synthetic pairs missed 3 of 5 required abstentions; then missed 3 and falsely abstained on 3 of 22 answerable requests; then abstained on all 5 but falsely on 10. Diagnosis found a train/serve precision mismatch that flipped learned abstentions and a lexical cue learned from unpaired negative examples. A deterministic resolver that binds a subject only from an identifier written in the request or from a 10-name curated table reached 1.0 on every metric of the same 27-item yardstick, with no false abstentions, using the first adapter unchanged; a model-free check confirms its binding on all 62 test prompts. A document-to-fields model was schema-valid on all 240 responses it returned, with relation F1 0.07; 71.1% of its corpus's relation labels shared no content word with their own sentence, and a relation-heavy curriculum raised unwitnessed predictions from 8 to 84. An append-only ledger refuted both values a 7B model proposed for a protein's length against a live database receipt. Samples are small and each training round ran once; we claim only what these runs show.

1 Introduction

A verification floor accepts only what it can check. In our earlier work a typed, source-pinned store is the only author of facts [40], and the boundary that decides what counts as verified does not move when the model that proposes claims gets smaller [41]. That boundary answers one question well: *is this claim about entity e supported by the evidence?* It cannot answer a question that comes before it: *is e the entity the person meant?* When a request says "hemoglobin alpha" and a model turns it into a structured proposal, somebody has to choose the database identifier. If the model chooses a valid identifier for the wrong protein, the floor verifies the wrong protein correctly, and nothing downstream can tell.

That choice happens at a *seam*: the point where a model's output becomes an identifier or a fact inside the system. A small local model is a cheap proposer, and the obvious way to make its seam safe is to train it: teach it by example to bind names to identifiers, to abstain when nothing grounds a request, and to extract relations only when the sentence states them. We call these three behaviours *grounding* (§2). This paper reports what happened when we tried to train them in.

It did not work. We fine-tuned a 3-billion-parameter proposer three times on the same binding-and-abstention seam, and each round failed differently: the first learned the output format but not when to refuse, the second added refusals in the wrong places, the third refused ten answerable requests to stop missing five unanswerable ones. We fine-tuned a document-to-fields model of the same size on a large corpus and obtained perfect schema validity with a relation F1 of 0.07, then traced the cause to training labels that the sentences did not state. What closed the proposer's seam was not a better training recipe but a deterministic resolver outside the model: a subject may be bound only if the request itself contains

a valid identifier or a name from a 10-name curated table. On the same 27-item yardstick, with the first round's weights unchanged, it scored 1.0 on every metric and produced no false abstentions.

The comparison is lopsided on purpose. The task, once written down, was a lookup: a finite table of names, plus "abstain otherwise". The resolver implements that specification directly; the training rounds tried to approximate it with a model and missed in both directions. The finding is not that the resolver is clever. It is that we spent three training rounds trying to learn a table we already held.

We publish these as negative results because each failure was specific enough to be useful: a precision mismatch between training and serving that flipped a learned boundary, a lexical cue learned from negative examples with no positive counterpart, a trade between missed and false refusals that three rounds moved along but never escaped, and a corpus whose relation labels were mostly authored from context the sentence did not contain. We state each as far as the evidence carries it and no further. Nothing here shows that fine-tuning cannot teach grounding in general; it shows what these runs did, at this scale, with these data.

1.1 Contributions

C1. A three-round negative result on training binding and abstention into a 3B model, with mechanisms (§3). Missed and false abstentions on a fixed 27-item yardstick moved from 3/0 to 3/3 to 0/10 across three LoRA rounds. We report the two diagnosed causes, the evidence behind each and its limits, and an audit of our own measurement: a denominator artefact in the abstention metric, a train/evaluation leak of 4 of 27 prompts, and a "hard set" that shares 23 of its 35 prompts with the first round's training and validation splits.

C2. A deterministic resolver that closes the binding seam on the same yardstick, and a model-free account of what it closes (§4). With the first round's weights, parse, kind, subject, abstention recall and admission all reach 1.0 with zero false abstentions. A model-free analysis over all 62 test prompts shows which half of the seam the resolver closes by construction (a confident proposal about an ungrounded subject cannot survive it) and which half stays with the model (it cannot undo a false abstention).

C3. A negative result on training relation extraction from labels the sentence does not witness (§5). 71.1% of 74,498 relation labels in the corpus fail even a lenient lexical witness test. After filtering, the model was schema-valid on every response and scored a relation F1 of 0.07; a relation-heavy curriculum raised the model's unwitnessed predictions from 8 to 84 with flat F1 and was rejected. The general lesson we draw is narrow and, we think, durable: *type-safe is not grounded.*

C4. A value-level instance and a synthesis of placement (§§6–7). An append-only hypothesis ledger refuted both values a local 7B model proposed for a protein's length against a live database receipt. We add an informal observation from a free-form tool-using loop, and we connect all four to the placement rule of our Inference Placement paper [39]: information that is already known is retrieved, not inferred.

What we claim, and what we do not

We claim that in these runs, at 3B parameters with a few hundred synthetic pairs (the proposer) or about 34,000 weakly labelled pairs (the document model), fine-tuning did not install the grounding behaviours we targeted, and that placing the grounding step in a deterministic component outside the model closed the proposer's seam on the same yardstick. We do not claim that fine-tuning cannot learn grounding: larger models, more data, other objectives or other training methods may. Every training round ran once, with one seed; the yardsticks have 27, 35 and 247 items; differences of one to three items are not statistically meaningful on their own, and we say so where they appear. No model was trained or run for this paper; every number comes from reports, logs and ledgers written at the time, and from model-free re-analyses listed in Appendix A.

Relation to earlier Perslis work. This paper is the training-side companion of *Inference Placement* [39], which asked where learned inference earns authority and answered with a measured map; here we measure what it costs to put a lookup inside a model anyway. It reuses the verification floor of *Peel* [40] and the separation of model-owned and runtime-owned capabilities from *The Verification Floor* [41], and it keeps its failures as first-class evidence in the manner of *Fail-First Models* [38]. We cite rather than repeat their results.

Paper map. §2 defines the three behaviours and the seam. §3 reports the proposer's three training rounds, §4 the resolver and the audit of our yardstick, §5 the document-to-fields model, §6 the ledger run and the informal tool-use observation. §7 draws the placement lesson, §8 places the work, §9 lists threats to validity.

2 Setting and terms

2.1 Three grounding behaviours

We use *grounding* for three behaviours a proposer needs before its output can be trusted as input to a verifier. Each is defined by what a correct system does, not by how it does it.

Definition 1 (Binding)**.** Mapping a mention in a request to an identifier in a reference database. Here the mentions are protein names or identifiers and the database is UniProt [30]; a binding is correct when the identifier denotes the protein the request names.

Definition 2 (Abstention)**.** Declining to propose when nothing in the request, or in the reference the system holds, grounds a binding: a fictional protein, a real protein outside the reference, an off-topic question, an ambiguous reference ("the protein from yesterday's meeting"), or a malformed identifier.

Definition 3 (Witnessing)**.** A relation extracted from a sentence is *witnessed* when the sentence states it. Witnessing is a semantic property we cannot measure directly at scale. We measure a necessary lexical proxy: an item passes when at least one of its content words (four or more letters, not a stopword) appears in the sentence or its topic. An item that fails the proxy is certainly unwitnessed; an item that passes may still be unwitnessed.

2.2 The seam, and the two ways to fail at it

A *seam* is the point where a model's output becomes an identifier or a fact inside the system. In the proposer of §3, the seam is the `subject` field of a structured proposal. Downstream of the seam sits the floor's admission step. We describe it only by its behaviour: it parses the proposal against a strict grammar and routes it to claim verification, evidence gathering, or a plan-only preview; it rejects malformed proposals; and it cannot tell a valid identifier for the wrong protein from the right one. Its implementation is withheld, as in our earlier papers [40].

That leaves two ways to fail at the seam, and they are not symmetric.

- A **missed abstention** is a confident proposal on a request that grounds nothing. If it is well-formed it is admitted, and the floor then verifies, gathers evidence about, or plans for a protein the person never named. This is the failure the floor cannot catch.
- A **false abstention** is a refusal of a request that the reference does ground. It costs coverage, not truth: the system says "I can't" when it could have.

A third outcome, an **unparseable reply**, is rejected by the strict grammar before admission and is harmless in the same sense as a false abstention. Figure 1 counts all three.

2.3 Evidence discipline

No model was trained, served or queried for this paper. Every number comes from an evaluation report, training log, ledger or run log written when the work was done, and every such number is listed with its file and line in the paper's claims ledger (`CLAIMS.md`). Where we could re-derive something without a model we did, read-only, on 27 September 2026: the resolver analysis of §4.3, the resolver's regression tests, the dataset audits of §4.5 and §5.2, the ledger's hash chain in §6, and two UniProt lengths. Where the only record of a result is a message written at the time and not a saved output, we say so.

3 Case study 1: training a proposer to bind and abstain

3.1 Task, data and yardstick

The proposer sits on top of the protein-science floor described in our earlier papers [40, 41]. For each request it must emit exactly one JSON object of one of four kinds: a *claim* about a protein's structural consistency, an *extract* asking the floor to gather evidence for an identifier it lacks, an *intent* proposing a laboratory operation with parameters (plan-only; nothing is dispatched), or an *abstain* with a reason. The subject of the first three kinds must be a bare UniProt accession. The prompt tells the model to use an accession for a named protein only when it is common knowledge and otherwise to abstain, because "a wrong accession is worse than no proposal". Replies are decoded at temperature 0.

The binding the proposer needs is small and fixed. A curated table maps 10 names to 8 human proteins (two proteins have two names each, for example "hemoglobin alpha" and "hemoglobin subunit alpha" for P69905). Everything else must be refused unless the request contains a valid accession outright.

Training pairs were generated from the floor by code, never written by hand: claims from the curated table, extracts for accessions the floor does not hold, intents with parameters drawn from the planner's valid ranges, and abstentions from fictional names, near-miss names that are real but not in the table ("hemoglobin gamma", "insulin receptor", "cytochrome c oxidase"), off-topic questions, ambiguous references and, from the second round, malformed identifiers. Every assistant turn is checked at build time against the same grammar the floor enforces, so the data cannot teach a shape the floor rejects.

The **yardstick** is the 27-item held-out split of the first dataset: 13 claims, 6 intents, 3 extracts and 5 abstentions. The second and third datasets were built with those 27 prompts excluded, so every round is scored on the same items. The evaluator reports six numbers, and their denominators matter (§4.5):

- *parse rate*: replies that satisfy the strict grammar, of all 27;
- *kind accuracy*: replies of the correct kind, of all 27 (an unparseable reply counts as wrong);
- *subject accuracy*: correct kind and correct accession, of the items whose gold is not an abstention;
- *abstain recall*: abstentions among gold-abstain items *that parsed*;
- *false abstentions*: abstentions on items whose gold is not an abstention (a count);
- *admission rate*: parsed proposals the floor processed without error, of those that parsed.

3.2 Three training rounds

All three rounds fine-tuned Llama-3.2-3B-Instruct [10] with LoRA [11] in the MLX framework [22]: rank 8, scale 20, no dropout, 16 adapted layers, learning rate 10^{-5}, batch size 2, 6.947M trainable parameters of 3,212.75M (0.216%). For serving, each adapter was fused into the bf16 base weights, converted with llama.cpp's converter [19] to an 8-bit GGUF file and served locally through Ollama [23]. Table 1 lists what changed between rounds.

Round v3 was configured for 400 iterations. At iteration 300 the external drive refused the checkpoint write and training stopped with an error; the iteration-200 checkpoint, whose validation loss (0.021) had

Table 1. The three training rounds. Train and valid are pair counts; abstain share is of the training split; validation loss is the last value logged for the checkpoint that was served.

Round	Training base	Train / valid	Abstain share	Iterations (val. loss)	What changed
v1	4-bit	135 / 18	11.1%	300 (0.017)	first dataset: 180 pairs, 5 fictional names
v2	4-bit	172 / 22	47.7%	400 (0.024)	near-miss and malformed-identifier abstentions; wider fictional, off-topic and ambiguous pools
v3	bf16	178 / 23	42.7%	200 (0.021)	trained on the bf16 base (native fuse); claim "contrast pairs" for every curated accession; near-miss templates cut from three to two

Table 2. Proposer results. Columns 1–5: the 27-item yardstick. Column 6: the 35-item "hard set" (§4.5). Rates are as reported; counts in parentheses are recovered from the rates and the evaluator's denominators (CLAIMS.md C1.30).

	prompt-only base model	v1	v2	v3	v1 + resolver (27 items)	v1 + resolver (hard set, 35)
Parse rate	0.741 (20/27)	1.0 (27/27)	0.963 (26/27)	0.963 (26/27)	**1.0** (27/27)	0.971 (34/35)
Kind accuracy	0.63 (17/27)	0.889 (24/27)	0.741 (20/27)	0.593 (16/27)	**1.0** (27/27)	0.971 (34/35)
Subject accuracy	0.304 (7/23)	0.955 (21/22)	0.818 (18/22)	0.5 (11/22)	**1.0** (22/22)	0.962 (25/26)
Abstain recall	0.5 (2/4)	0.4 (2/5)	0.4 (2/5)	1.0 (5/5)	**1.0** (5/5)	1.0 (9/9)
False abstentions	1	0	3	10	**0**	0
Admission rate	1.0 (20/20)	1.0 (27/27)	1.0 (26/26)	1.0 (26/26)	**1.0** (27/27)	1.0 (34/34)

flattened, was served. All three rounds reached low validation loss (0.017–0.024 from starting values near 3.4): each learned its training data. The question is what that data taught.

3.3 Results

v1: the format was learned, the boundary was not. The first round did what format training does. Unparseable replies fell from 7 to 0, kind accuracy rose from 0.63 to 0.889 and subject accuracy from 0.304 to 0.955. The prompt-only base model had, for example, answered a question about cytochrome c with the well-formed accession `P0A9M2` and an invented claim shape; the floor rejected the reply because of the shape, not the accession (the curated table maps cytochrome c to `P99999`). After training the shapes were right. Abstention was not. The recipe's own ship bar required v1 to beat the base model on abstain recall, and the report records a fall from 0.5 to 0.4. That fall is a change of denominator, not of behaviour: the base model abstained on 2 of the 4 gold-abstain prompts it answered in grammar (its reply to the fifth was unparseable and rejected), and v1 parsed all 5 and abstained on the same number, 2. What training did was turn one harmless, rejected reply into a well-formed confident proposal. All 27 of v1's parsed proposals were admitted, including the three confident proposals on requests that ground nothing.

v2: more refusals in the data, not in the right places. The second round raised the abstention share of the training split from 11.1% to 47.7% and added the dangerous classes. On the yardstick it still missed 3 of 5 abstentions and now also refused 3 answerable requests. Two causes were diagnosed at the time.

(a) *Precision mismatch.* The adapter had been trained against a 4-bit quantized base, as in QLoRA [4], and was served fused into the bf16 base and then converted to 8-bit. An A/B run then showed that the adapter on its own 4-bit base abstained correctly on the yardstick's abstention items, while the served model turned them into confident claims. The training recipe had called this mismatch "mild" and "acceptable for a format-teaching LoRA"; for a decision boundary it was not. Two cautions apply. The raw outputs of the A/B were not saved; the record is the commit message written when it was run.

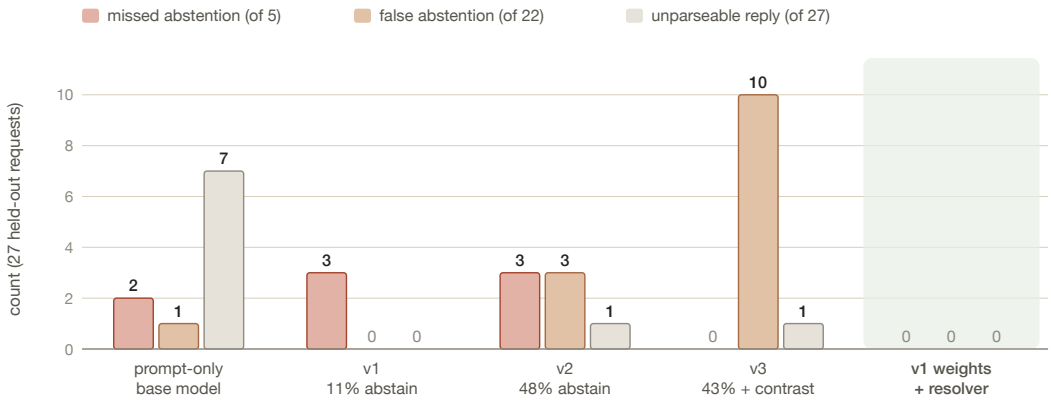

Figure 1. Where each round's errors went, on the 27-item yardstick. Missed abstentions are confident proposals on the 5 requests that ground nothing; false abstentions are refusals of the 22 answerable requests; unparseable replies are rejected before admission. Training moved the errors from one bar to the other and never removed both. The resolver (shaded) is the deployed combination with the v1 weights.

And the comparison changed more than precision: it also changed the runtime (training framework against serving runtime) and added the final 8-bit conversion. Precision is the most direct difference, but the A/B does not isolate it.

(b) *A lexical cue from unpaired negatives.* The new malformed-identifier abstentions all used the phrasing "accession ⟨*id*⟩", and no valid identifier appeared in that phrasing in the training data. The model learned the word, not the rule: it refused curated names and the bare accession P69905. This was found by inspecting the failing items, not by an A/B.

v3: the other side of the seesaw. The third round addressed both diagnoses and one more worry at once: it trained on the bf16 base so the fuse was native, added claim "contrast pairs" in the "accession ⟨*id*⟩" phrasing for every curated accession, and cut the near-miss templates from three to two because near-miss density had seemed to bleed into false refusals. It abstained on all 5 abstention items, and refused 10 of the 22 answerable ones; kind accuracy fell to 0.593 and subject accuracy to 0.5. Because three changes were made together, v3 cannot attribute the over-correction to any one of them. It does show that removing the precision mismatch did not, by itself, produce a correct boundary.

The pattern. Across the three rounds, missed and false abstentions went from 3/0 to 3/3 to 0/10, and total seam errors from 3 to 6 to 10 (Figure 1). Each round moved the operating point; none reached zero on both. We return to why in §4.6.

4 The table that closed the seam

4.1 What the resolver does

After the strict parse, every proposal that is not an abstention passes through a deterministic resolver before the floor sees it (Figure 2). The resolver computes the set of identifiers the request itself grounds, from two sources only:

(i) identifiers written literally in the request that pass the same identifier validator the floor uses; and

(ii) matches of curated names from the 10-name table, as whole words, longest name first, and void when

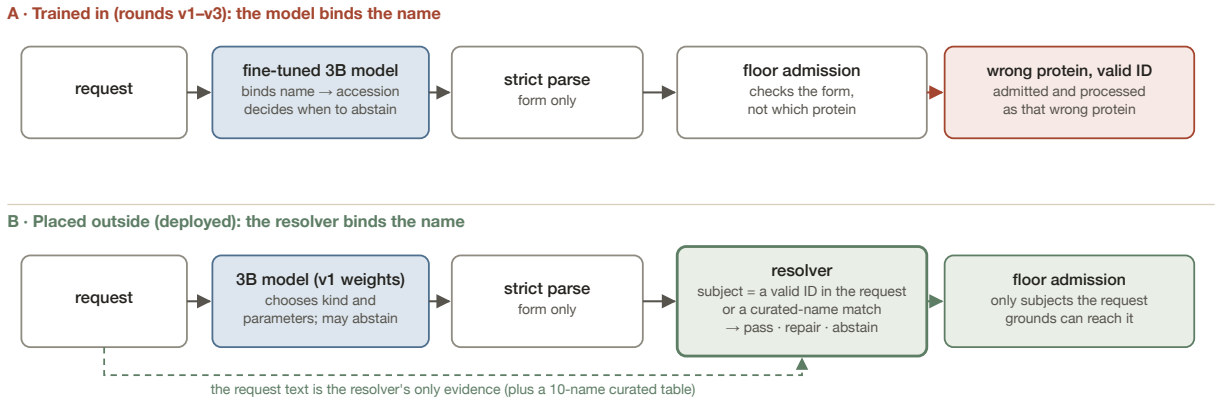

Figure 2. Where the binding lives. A: in the three training rounds the model chose the identifier, and the floor's admission step, which checks form, admitted whatever well-formed identifier it chose. B: in the deployed lane the model still chooses the proposal kind and its parameters, but the subject must come from the request text or the curated table. The resolver's only evidence is the request itself.

> the next word extends the name into a different protein: "cytochrome c oxidase" never binds cytochrome c, "insulin receptor" never binds insulin, and a fused form such as "proinsulin" does not match at all.

It then does one of three things. If the proposal's subject is in the grounded set, it passes. If exactly one identifier is grounded and the model named another, the subject is *repaired* to the grounded one. If nothing is grounded, or several are and the model named none of them, the proposal is replaced by an abstention. An abstention from the model passes through untouched. The resolver never consults the model's recall of accessions, and it never adds a binding the request does not justify. The implementation is small and not the subject of this paper; we describe it only by this behaviour.

4.2 Result on the same yardstick

With the v1 weights unchanged, the combination scored 1.0 on parse rate, kind accuracy, subject accuracy, abstain recall and admission rate on the 27-item yardstick, with no false abstentions (Table 2, fifth column). On the 35-item hard set it abstained on all 9 abstention items with no false abstentions; one reply, an intent missing a required field, was rejected as unparseable, the harmless outcome, which is why parse, kind and subject accuracy read 0.971, 0.971 and 0.962 there. No new weights were promoted. The v2 and v3 models were kept as forensic evidence and are not served.

4.3 What the resolver closes by construction, and what it does not

The gate result above depends on a model. Part of it does not, and we can show which part without running one. We ran the resolver's grounding step, read-only and without any model, over every prompt of both evaluation sets, at the commit that produced the reports and at the current head (a later review extended the list of words that void a name match). Table 3 gives the result.

Two consequences follow for *any* model whose reply parses. On every item that should be refused, a confident proposal is converted to an abstention, because nothing is grounded. On every answerable item, a proposal of a non-abstention kind leaves the resolver with the gold identifier, by pass or by repair. That is the half of the seam the resolver closes by construction: a confident proposal about an ungrounded subject cannot reach the floor. The resolver's 42 regression tests (37 when the reports were produced, 5 added by the later review), one per curated name, one per near-miss name, one per blocked extension phrase and

Table 3. Model-free analysis of the resolver over all 62 test prompts, run on 27 September 2026. "Grounded" is the set of identifiers the resolver derives from the request text. The worst-case column feeds the resolver a claim naming a wrong, well-formed identifier for every prompt. Results are identical at both resolver versions.

Set	Prompts	Gold abstain	grounded set is empty	Answerable	grounded set is exactly the gold identifier	Worst case handled
Yardstick	27	5	5 of 5	22	22 of 22	27 of 27
Hard set	35	9	9 of 9	26	26 of 26	35 of 35

one per pass, repair and abstain case, all pass on a copy of the current code.

The other half stays with the model. The resolver cannot undo a false abstention, because it passes abstentions through; it cannot fix a wrong choice among claim, extract and intent; and it cannot parse a malformed reply. The v1 weights made no false abstentions on either set, and that is why the combination reaches 1.0. Paired with the v3 weights, the same resolver would keep v3's ten false abstentions; that follows from the construction and was not measured. Choosing the proposer's weights still matters, for the part of the task that is not a lookup.

The resolver also has a documented trade-off, pinned by a test: if a request grounds exactly one curated protein while the person meant another, unnamed one, repair binds the one that was named. We accept this because the alternative, trusting an identifier the request does not ground, is the failure the resolver exists to remove. Its coverage is the table: any protein not in it is refused unless the request gives its accession. Extending coverage means adding reviewed rows to the table, not retraining.

4.4 The comparison is lopsided by design

The yardstick's gold labels were generated from the same curated table that the resolver consults, so the resolver's agreement with gold is, on these prompts, a consequence of its construction; Table 3 shows it directly. We do not present this as the resolver winning a fair race. We present it as the finding: once the task was written down precisely, it was a finite lookup plus "refuse otherwise", small enough to implement exactly in the runtime. The three training rounds tried to approximate that lookup inside a model and missed in both directions. This is the "known information is retrieved, not inferred" rule of our Inference Placement paper [39], reached here from the other side, by paying for the alternative first.

4.5 Auditing our own yardstick

Reading the evaluation files for this paper turned up three problems in our own measurement. None changes a conclusion, and we report all three.

A denominator artefact. Abstain recall is computed over gold-abstain items *that parsed*. The prompt-only base model's 0.5 is 2 of 4 and v1's 0.4 is 2 of 5: the same two correct abstentions (§3). The recorded "regression" was real as a statement about admitted confident proposals, not as a statement about correct refusals. Subject accuracy has a related quirk: its denominator excludes gold-abstain items that parsed but still counts unparseable ones, which is why the base model's denominator is 23 and the others' 22.

A train/evaluation leak. The dataset generator emitted 10 duplicate prompts: two curated names map to each of P69905 and P04637, so the templates that use the bare accession were produced twice. Four of the 27 yardstick prompts appear verbatim, with identical gold, in v1's training split. All four are claims about a bare accession. They cannot affect any abstention result, and at most they flatter v1's claim metrics by up to four items.

The "hard set" is v3's held-out split. The 35 items were held out from v3's training, not from v1's. Twenty-two of them appear verbatim in v1's training split and one more in its validation split. Six of its nine abstention items are new to v1 (three near-miss names, two new fictional names, one new off-topic question); three were in v1's training data. Despite being described at the time as containing near-miss *and malformed* classes, it contains no malformed-identifier item. The hard-set numbers in Table 2 should therefore be read as a check that the combination does not break on new near-miss and fictional names, not as a clean held-out test of the v1 weights. The model-free result in Table 3 does not depend on the model and is unaffected.

4.6 Why this seam resisted training

We offer an explanation as a hypothesis, with the evidence for it and the evidence it lacks.

The specification is a membership test over a tiny positive set (10 names and any literal valid accession) against an unbounded negative set whose members can differ from a positive by one word: "insulin" and "insulin receptor", "hemoglobin alpha" and "hemoglobin gamma". The base model's prior supplies a confident, well-formed accession for almost any protein-like name, as the prompt-only model's answer for cytochrome c shows. With 135 to 178 examples, a 3B model does not acquire the membership test; it acquires a score over surface features of those examples, and the threshold on that score moves with the data mix, with a lexical cue that happens to correlate with the label (v2), and, v2 suggests, with the numeric precision of the weights.

> **Hypothesis H1.** At this scale and with training sets of this size, a small closed whitelist is not learnable as a decision boundary: moving the model's implicit threshold trades missed abstentions for false ones, and no setting removes both.

The evidence for H1 is three single-seed rounds whose errors moved from 3/0 to 3/3 to 0/10, with no round at 0/0. It lacks a seed sweep, a learning curve over training-set size, a comparison of training objectives, and a larger model. A training recipe at this scale that reached zero missed and zero false abstentions on both sets across several seeds would refute it. Even if H1 were refuted, we would keep the binding in the table: a table is exact, can be read and reviewed, and changes without retraining, and the three rounds bought nothing it does not give.

5 Case study 2: a document-to-fields model

5.1 The model and its labels

The second model turns one sentence of documentation into one structured record with a fixed schema: categorical labels, free-text fields, and six typed relation lists that should name what the sentence says the subject causes, requires, is part of, and so on. It serves an extraction lane whose design, schema and storage are withheld; we report only the model's evaluation and what it taught us.

The model is a QLoRA [4] fine-tune of Llama-3.2-3B-Instruct [10] on a 4-bit base: rank 16, dropout 0.05, learning rate 10^{-4}, batch size 2, sequences of up to 704 tokens (the longest measured example was 641), loss on the completion only, 13.894M trainable parameters (0.432%), 3,000 iterations. Validation loss fell from 1.764 to 0.562 (lowest 0.535, at iteration 2,400). For serving, the adapter was fused into the bf16 base, converted with llama.cpp's converter and quantized in Ollama to a 2.0 GB `q4_K_M` build.

The labels came from a corpus of earlier structured records written for sentences *while the surrounding conversation was in view*. We scanned 302,066 corpus rows (201,355 distinct sentences) and selected 18,321 sentences, split by sentence into 247 held-out test sentences, 384 for validation and the rest for training; relation-rich sentences were repeated in training, giving 33,710 training examples.

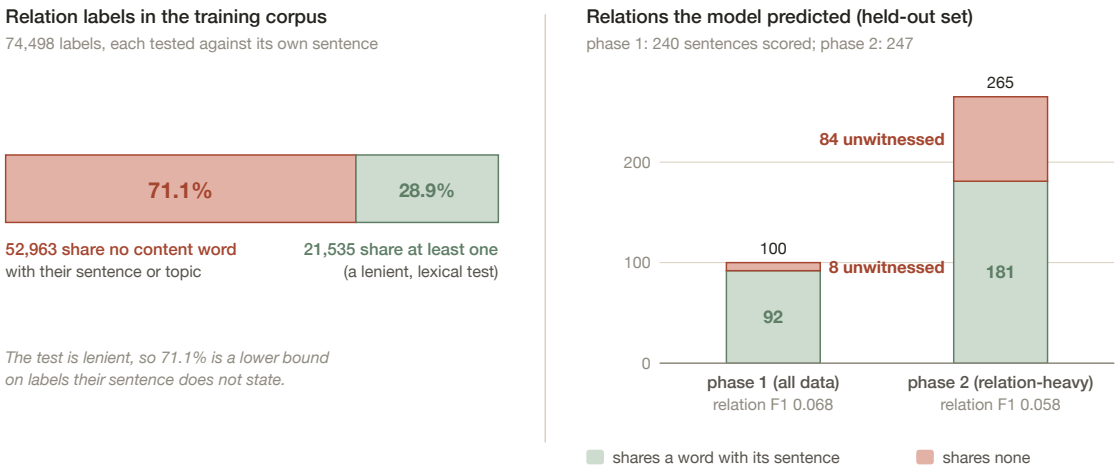

Figure 3. Left: the witness audit of the corpus's relation labels. Right: relation items the model predicted on the held-out set before and after a relation-heavy curriculum. The curriculum made the model say more; a smaller share of what it said shared even one word with the sentence, and relation F1 did not improve.

5.2 The witness audit

Before training, every relation item in the scanned rows was tested against its own sentence with the lexical witness proxy of §2. Of 74,498 relation items, **52,963 (71.1%) shared no content word with their sentence or topic** (Figure 3, left). These are not borderline cases: an item that fails the proxy names nothing the sentence mentions (the lane's code gives the example of a prerequisite naming a military campaign on a sentence that never mentions it). Because the proxy is lenient, 71.1% is a lower bound on the share of labels the sentence does not state.

Such labels were authored from context the model will never see at inference time. Trained on them, a model can only learn to produce relations from outside its input, which is hallucination by construction; the summarization and data-to-text literature documents the same effect when references contain content the source lacks [20, 5, 24]. The lane therefore dropped every unwitnessed item from the labels before training; the model was trained only on items that pass the proxy.

5.3 Phase 1: type-safe, and not grounded

The held-out evaluation sent all 247 test sentences through the same client and validator the live lane uses. Seven requests failed at the transport layer (the local model server returned errors) and were counted separately from quality; the other 240 returned a response. Table 4 and Figure 4 give the result.

Every response the model returned was valid against the schema, and 92 of its 100 predicted relation items shared at least one content word with the sentence. On content, it failed: categorical accuracy 0.633 against a gate of 0.85, and relation F1 0.068 against a gate of 0.60, with per-type F1 between 0.00 and 0.19. The verdict recorded at the time was "not usable" as a relation extractor.

The two passing numbers are weaker than they look. Schema validity says the output has the right shape; it says nothing about whether the relations are true, and grammar-constrained decoding [9, 32] would have given the shape for free. The witness share is a lexical proxy: the lane's own documentation later renamed it a "legacy lexical-overlap score" that "does not prove a relation's truth, direction, or polarity". A single anecdote from the post-export smoke test makes the point: for a sentence stating two explicit relations, the model returned a schema-valid record that restated one of them in a free-text field while both corresponding relation lists were empty. The output was type-safe. It was not grounded, and it was not complete. We draw the lesson in its narrowest form: *type-safe is not grounded.* A check of form,

Table 4. Document-to-fields model on the 247 held-out sentences. The gates were fixed before the evaluation. Categorical accuracy is the mean exact-match accuracy of two categorical fields; relation F1 is the mean over six relation types of a soft set-match F1 against single-reference gold; the witness share is the fraction of predicted relation items passing the lexical proxy.

	Phase 1 (all data)	Phase 2 (relation-heavy)	Gate
Sentences attempted / transport failures	247 / 7	247 / 0	–
Schema-valid responses	1.0 (240 of 240)	1.0 (247 of 247)	≥ 0.98
Categorical label accuracy	0.633	0.658	≥ 0.85
Relation F1 (six types; three types at 0.00 in both phases)	0.068	0.058	≥ 0.60
Predicted relation items	100	265	–
Witness share of predicted items	0.92 (92 of 100)	0.683 (181 of 265)	≥ 0.80
Unwitnessed predicted items	8	84	–
Mean token F1 over six free-text fields (derived)	0.29	0.38	–
Median latency per sentence	3.9 s	7.2 s	–
Verdict	not usable	not usable; rejected	

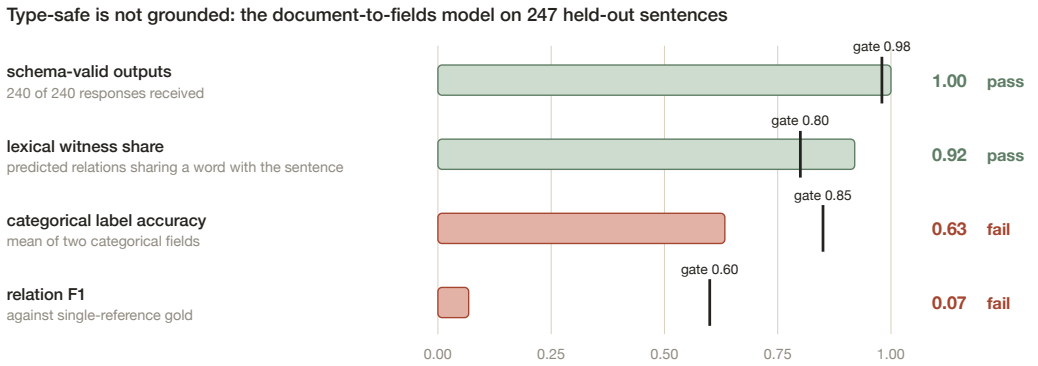

Figure 4. Phase-1 gates. Both checks of form passed; both checks of content failed. The witness share is itself only a lexical proxy.

however strict, is not a check of content.

5.4 Phase 2: a curriculum that made the model say more

The phase-1 model under-extracted: it predicted 100 relation items across 240 sentences with six lists each, so at least 93% of its relation lists were empty. The next attempt continued training from the phase-1 weights for 2,000 iterations on the 24,088 of 33,710 training examples that carry at least one relation, to break the prior toward empty lists. It worked, in the sense that the model predicted 265 relation items instead of 100, and several free-text fields improved (mean token F1 0.29 to 0.38). But the witness share fell from 0.92 to 0.683: unwitnessed predicted relations rose from 8 to 84 (Figure 3, right), and relation F1 stayed flat at 0.058. The curriculum was rejected and the phase-1 weights were kept. Pushing a model to produce more of a thing its labels did not reliably witness produced more of that thing from outside the sentence.

5.5 What limits these numbers

Single-reference gold. Each test sentence has one gold record; a different but defensible relation scores zero. The gold was also filtered with the same lexical proxy, so the evaluation labels are themselves only lexically witnessed. Relation F1 of 0.07 therefore bounds the model's agreement with a noisy label set, not

its accuracy in any absolute sense. The phase comparison is internally consistent (same 247 sentences, same scorer), which is why we lean on it.

Transport failures and the verdict rule. Phase 1 lost 7 of 247 requests to server errors. The rule at the time classed an evaluation as inconclusive only above 10% transport failures, so phase 1 was scored; the lane later adopted a stricter rule under which any transport failure makes the run inconclusive, and phase 1 would be inconclusive under it. Its content numbers are over the 240 responses received. Phase 2 had no transport failures. A 60-sentence spot check run after the final export is excluded: the lane's own rule makes a truncated run inconclusive.

Engineering failures, also negative results. Exporting the fused model to GGUF with the training framework's exporter failed twice: once with a serialization error ("can only serialize row-major arrays", logged), and once, after a workaround, with a runtime dtype assertion in the serving runtime (recorded in the export script). Conversion with llama.cpp's own converter worked and is the path the lane uses. During evaluation the local model server died more than once under memory pressure, which is why transport failures are counted apart from quality and why an evaluation with too many of them is reported as inconclusive, never as a score. A third continuation was prepared (audited data, frozen gold hash, recorded manifest) but never trained; there is no result to report.

5.6 What the lane does now

The lane is labelled as a pilot-grade enrichment of the non-relation fields, not a relation extractor. Before a record is stored, every predicted relation item is put through the same witness proxy and unwitnessed items are dropped. For documents, a pinned test requires that stored relations come from a deterministic source rather than from the model's output; how that source works is withheld. The model is still used; it is no longer trusted with the seam where a relation becomes a fact.

6 Case study 3: a ledger refutes a model's values

The first two case studies are about identifiers and relations. The third is about values, and it involves no training at all; we include it because it shows the same placement working one level up, where the thing a model gets wrong is a number.

6.1 Setup

A small hypothesis-testing loop runs *hypothesis, experiment, receipt, critic, blame, learning, new hypothesis.* Its state lives in an append-only ledger, and everything that decides what may be claimed is mechanical: the state machine, the verdict arithmetic over predictions registered before the receipt, the confidence rule and the blame step contain no model. A model may act only at one seam, the hypothesizer, where it must propose at least two competing hypotheses with predictions; the ledger records them through the same rule-gated interface a person would use, and a malformed or single-idea proposal is refused and records nothing.

6.2 The run

A local model, `mistral:7b-instruct` [14] served by Ollama, was asked how many residues are in the canonical UniProt sequence of human hemoglobin subunit alpha (P69905). Figure 5 shows the ledger it produced.

The model proposed two hypotheses: a length of 146, with the rationale that this is "the reported sequence length for P69905 according to the UniProt database", and a length of 147, with the rationale that N-terminal acetylation "would add an additional amino acid to the sequence". Both predictions were registered with an exact-equality relation and zero tolerance. The loop then designed one experiment, and its UniProt adapter recorded a receipt: an HTTP 200 response of 271,287 bytes from the UniProt REST interface, stored with the SHA-256 digest of the response body, observing a sequence length of 142. The critic scored 147 as refuted (error −5) and 146 as refuted (error −4), both hypotheses moved to REJECTED, and

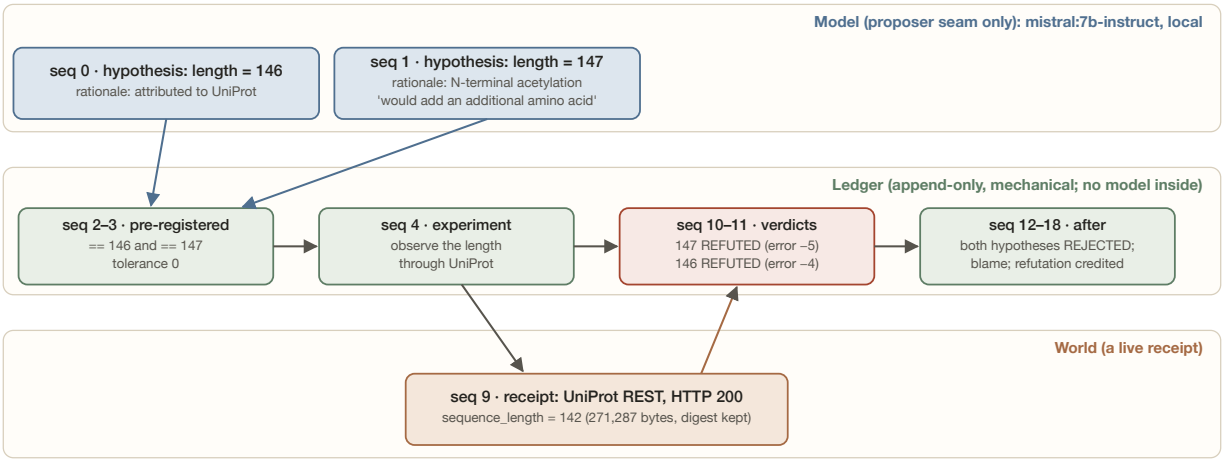

Figure 5. The ledger of the run, event by event. The model's two predictions were registered (events 2–3) before the receipt existed (event 9), so both were gradable; both were refuted by arithmetic against the receipt. The chain was re-verified read-only for this paper.

the loop credited knowledge for "a refutation across 2 hypothesis(es)". The 19 events form a SHA-256 hash chain; we recomputed every link read-only and the head matches its recorded sidecar. The chain makes in-place edits detectable to anyone holding the head; it is not signed and proves integrity, not authorship.

We re-read UniProt on 27 September 2026: P69905 still has length 142, and the beta chain, P68871, has length 147 [30]. One of the model's two values is the neighbouring protein's length; the other is the canonical length of neither chain. *The Verification Floor* reports the same shape of error from a small model on the same protein [41]; we do not repeat that analysis. Neither rationale survived: the database reports 142, not 146, and the length being asked about counts the residues of the canonical sequence, which a modification such as acetylation does not add to.

6.3 A quirk of mechanical blame

After the refutations, the blame step raised suspicion on the assumptions the model had stated and generated one follow-up hypothesis per refuted hypothesis. For the 147 hypothesis the follow-up is reasonable ("Assumption is false: The N-terminal valine… is acetylated"). For the 146 hypothesis it is not: because the model had claimed, falsely, that UniProt reports 146, the loop generated the hypothesis that "the information provided by the UniProt database is accurate" is false. The follow-up is only a hypothesis; it would need its own experiment, and nothing entered the ledger as fact. But it shows a real limitation: mechanical blame can only blame what the proposer said, so it inherits the proposer's framing, including a misattribution.

Scope. This is one run of one question with one model. It is an existence proof that the ledger disposes of what the model proposes, not a measurement of the model. The run's ledger is kept on disk and deliberately not committed to the project's repository, whose policy is to commit only the canonical example run; its integrity rests on the hash chain and on the commit message written when it ran.

6.4 An informal observation: free-form tool use

We add one observation that is not a training result and is weaker than everything above. In a free-form, tool-using agent loop, seven runs of the same biomedical question ("What evidence connects BRCA1 dys-

Table 5. The seven recorded free-form runs. "Actions" counts tool calls started. No run called a biomedical database connector or the symbolic floor.

Model	Actions	What it did
hosted model via CLI	0	provider returned a usage-limit error, fallback rate-limited; no action
`qwen2.5-coder:7b` (local)	0	no action or completion recorded
Claude via CLI	1	one web search, then a structured answer listing eight journal references, none retrieved through a database connector in that run
`qwen2.5-coder:7b`	17	alternated between asking the (absent) user and searching the code base for the literal text `your_search_pattern_here`, 8 times
`qwen2.5-coder:7b`	8	read a repository instructions file 4 times, asked the user once, one web search, one failed read, a denied file write, then a three-sentence uncited answer
`qwen2.5-coder:7b`	5	asked the user twice, one web search for "code investigation techniques", two failed file reads; ended planning to read a source file
`qwen2.5-coder:7b`	4	asked the user once, searched for and read repository recovery instructions; ended summarising them

function to breast cancer? Cite sources.", one run with a longer variant) were recorded on 20 September 2026. Table 5 summarises the preserved run logs.

Across the 35 recorded actions there were 3 web searches, 9 file reads, 8 code searches, 1 context search, 1 attempted write and 13 attempts to ask the user; there were no calls to the biomedical connectors or the floor. In three of the four runs in which the 7B coder model [12] acted, it treated the question as a task about the code base; in the fourth it read a repository file four times before searching the web.

Two cautions make this an observation rather than a result. First, the logs do not record which tools each run exposed to the model. The runs were made while the loop's science configuration was being changed; the project-level configuration that declares the biomedical connector was committed after these runs, and the user-level configuration as it stood that day was not preserved, so we cannot show that the connector was available in every run. Second, the change log written that day also records that removing web search led a capable model to fabricate citations from memory; no transcript of that run was kept, so we report it as unverified. What the runtime did in response is behaviour we can state: the grounded research tools were made reachable as deterministic commands that do not depend on the model choosing them, alongside the free-form loop.

7 Discussion: where the grounding step lives

7.1 Four seams, one placement

Table 6 puts the case studies side by side. In each, something had to decide whether a model's output could become an identifier, a relation or a fact. Where that decision was trained into the model, it failed in a way we could measure. Where it was placed in a deterministic component that consults only the request and a curated reference, the seam held on the same measurement.

7.2 Relation to Inference Placement

Inference Placement [39] proposed a rule for allocating computation by the epistemic status of the information: what is already known is retrieved, what is unknown but measurable is measured, what needs verification goes to the floor, and only genuine uncertainty is a candidate for learned inference, which must then earn its place. It also gave three gates for granting a learned component authority: theoretical value, realizable recovery at a supportable data price, and an adequate model class.

The binding seam fails the first gate by construction. For a name in the curated table, the correct

Table 6. The seams, what training attempted, how it failed, and what closed them.

Seam	What we trained or tried	How it failed	What closed it	Still open
Name → identifier (§3)	three LoRA rounds on 135–178 synthetic pairs	missed/false abstentions 3/0, 3/3, 0/10	resolver: identifier in the request or curated name, else abstain	coverage is the 10-name table
Refusing ungrounded requests (§3)	abstention-heavy data; contrast pairs	a lexical cue; over-refusal	forced abstention when nothing is grounded	false abstentions remain the model's
Witnessed relations (§5)	fine-tuning on filtered labels; a relation-heavy curriculum	F1 0.07; unwitnessed predictions 8 → 84	witness filter before storage; deterministic source for document relations	relation extraction itself
Values (§6)	none: a 7B model proposed	both values wrong	pre-registered prediction against a receipt	blame inherits the proposer's framing
Choosing grounded tools (§6.4)	none: instruction only (informal)	no connector calls in seven runs	grounded tools also offered as deterministic commands (not measured here)	availability unrecorded

identifier is already known; for any other name, the specification says to refuse; there is nothing left for a learned binding to add over the table. Inference Placement predicted that the binding belongs in retrieval. This paper supplies the other half of the argument: what happens when you put it in the model anyway. The answer, on this seam, is three rounds of errors in both directions.

Relation extraction is different in kind. It is not a lookup, and it may have real value for a learned component. What failed there was the data: labels written from context the input does not contain cannot teach grounded extraction. That is a precondition for the second gate (recovery at a supportable data price), and our corpus did not meet it.

7.3 Relation to the Verification Floor

The Verification Floor [41] separated capabilities that belong to the model (navigating evidence, knowing when to stop) from one that belongs to the runtime (what may count as verified), and found that the runtime-owned floor did not move as models shrank. The proposer's seam shows why binding belongs on the runtime side of that line. Binding is not navigation: getting it wrong does not slow the system down, it points verification at the wrong entity, and verification then succeeds. We moved the part of abstention that is really a lookup ("is this name in the table?") out of the model and left the part that is judgement ("is this request about proteins at all, and which kind of proposal fits it?") with the model, which on these items did it well.

7.4 Type-safe is not grounded

Both training case studies had solved form before they failed on content: the proposer's parse rate was 1.0 after one round, and the document model was schema-valid on every response it returned. Constrained decoding would have delivered that form without training [9, 32]. Form is necessary for a verifier to read a proposal and says nothing about whether the proposal is about the right thing. A system that reports schema validity, or a lexical overlap score, as evidence of grounding is reporting the wrong quantity. We would put it as a checklist item: for every seam, name the check that tests content rather than form, and if there is none, the seam is not grounded.

7.5 Why negative results on small models matter here

Small local models are attractive for exactly the settings where grounding matters most: offline, private, on-device [40]. It is tempting to treat their weakness as a training problem to be solved before deployment. These runs suggest a different division of labour. Let the small model do what it did well here, choosing the kind of proposal and its parameters, and give the grounding steps to components that can be exact. The model is still useful; it is not trusted with the seam.

8 Related work

Hallucination, abstention and over-refusal. Hallucination is usually defined as generated content that is nonsensical or unfaithful to its source [13]. Two theoretical results frame our seesaw. Calibrated language models must hallucinate certain kinds of arbitrary facts at a rate tied to how many of them appear only once in training [15]; and under binary grading abstaining is strictly suboptimal, so training and evaluation reward guessing over acknowledging uncertainty [16]. Refusal-aware instruction tuning teaches a model to decline questions beyond its parametric knowledge [36]. Later work reports that such tuning can make models refuse questions they could have answered correctly, which it names over-refusal [37], and that direct fine-tuning for noncompliance can produce over-refusal while parameter-efficient adapters strike a better balance [3]; a survey of abstention notes that supervised fine-tuning can make models more conservative [31]. XSTest shows refusals triggered by surface words rather than meaning [27]; our v2 cue (the word "accession" leading to refusal) is a small instance of the same thing, and our three rounds a small instance of the trade-off. What we add is a case in which the thing to be refused is defined by a closed list the system already holds, so the trade-off can be removed rather than tuned.

Learning from labels the input does not support. Fine-tuning on examples that introduce knowledge the model lacks increases its tendency to hallucinate [8]. In summarization, human annotators found content not supported by the source document in 76.9% of a sample of XSum reference summaries, and models trained on such data are prone to unfaithful output [20]. In table-to-text generation, 62% of the WikiBio references examined mention information not in the table, which misleads reference-based metrics [5]; ToTTo had annotators delete unsupported phrases from references for this reason [24]. Distant supervision for relation extraction treats every sentence in which an entity pair co-occurs as evidence of their knowledge-base relation [21]; that assumption was measured failing for about 31% of cases on news text, which motivated the expressed-at-least-once assumption [26]. Deep networks readily fit labels unrelated to their inputs [35], so an unwitnessed label is learned, not rejected. The 71.1% in our corpus is a stronger violation than distant supervision's: the failing labels share no content word with their sentence at all. Filtering labels by a witness test before training is the move ToTTo made by hand; our negative result is that, with single-reference labels filtered only lexically, it did not yield a usable relation extractor at 3B.

Quantized fine-tuning. LoRA trains low-rank updates on a frozen base [11]; QLoRA backpropagates through a frozen 4-bit base into LoRA adapters [4]. LoftQ reports a consistent gap between full fine-tuning and quantization plus LoRA, and attributes it to the discrepancy between quantized and full-precision weights [18]. QA-LoRA observes that QLoRA's adapters are merged back into 16-bit weights and that post-training re-quantization loses accuracy at low bit widths [33]. Our v2 diagnosis is a practitioner's instance of this interaction, on a decision boundary: an adapter trained against a 4-bit base behaved differently once fused into bf16 and converted to 8-bit. We tried neither remedy.

Structured output. Grammar-constrained decoding guarantees that output follows a given structure without fine-tuning [9]; guided generation does the same with regular expressions and context-free gram-

mars [32]. Both of our training case studies had already achieved perfect form. Their failures were in content, which these methods do not address and do not claim to.

Entity linking with curated vocabularies. Mapping text to a controlled vocabulary by knowledge-intensive symbolic matching has a long history in biomedicine; MetaMap maps biomedical text to the UMLS Metathesaurus [1]. Curated databases such as UniProt define the identifiers [30]. Our resolver is a deliberately tiny instance of this, and we claim nothing new about how it matches; the point is where it sits.

Retrieval, tools and citations. Retrieval-augmented generation conditions a generator on retrieved passages but leaves the generator the author of the final text [17]. Self-RAG trains a model to decide when to retrieve and to critique its own output with reflection tokens [2]: a way of training grounding behaviour into a model, at larger scale and with different supervision than ours. Toolformer and ReAct let models decide when to call tools [29, 34]; Gorilla documents hallucinated API calls and reduces them with retrieval-aware training [25]; ALCE finds that even the best models' citations often lack complete support [6]. Our informal tool-use observation is consistent with this line and adds only a small, candid record.

Contamination. Contamination of evaluation data leads to overestimated performance [28]. Our own yardstick had a leak of 4 of 27 prompts, which we found only while writing this paper (§4.5).

Division of labour. Neurosymbolic work argues for combining learned and symbolic components [7]. Our earlier papers took a specific position within it: a symbolic store as the only author of facts [40], a verification floor owned by the runtime [41], a measured placement of learned inference [39], and learning from verified failures as explicit rules [38]. This paper adds the negative control those positions implied: what happened when we moved a grounding step into the model instead.

9 Limitations and threats to validity

These are stated so the results cannot be over-read.

- **Small samples, one seed.** The proposer's yardstick has 27 items (5 of them abstentions) and its hard set 35; the document model's test set has 247 sentences. Each training round ran once, with one seed. Differences of one to three items between rounds are within what a different seed could produce; we rely on the pattern across rounds, not on any single difference, and we report counts beside every rate.
- **The metrics have quirks, and we found some of them late.** Abstain recall is computed over parsed items, so the recorded fall from 0.5 to 0.4 is a denominator change (§4.5). Four yardstick prompts leaked into v1's training split; the hard set shares 23 of 35 prompts with v1's training and validation splits and contains no malformed-identifier item despite being described that way.
- **One diagnosis rests on a record, not a saved output.** The A/B that implicated the precision mismatch in v2 is known from the commit message written when it was run; its raw outputs were not kept, and it changed the runtime and the final 8-bit conversion as well as the precision. The second v2 diagnosis came from inspecting failing items, not from an A/B.
- **v3 changed three things at once** (training precision, contrast pairs, template mix) and was served from its iteration-200 checkpoint after a storage error stopped training at iteration 300. It cannot attribute its over-refusal to any one change.

- **The resolver's success is partly by construction.** The yardstick's gold was generated from the curated table the resolver uses (§4); agreement with gold on these prompts is guaranteed for the binding half. The model-dependent half (false abstentions, kind choice, parsing) was measured on 27 and 35 items only. The resolver's coverage is 10 names, and its repair rule binds a protein the request names when the person may have meant another, unnamed one; that trade-off is deliberate and pinned by a test.
- **The document model's metrics are weak proxies.** Relation F1 is measured against single-reference gold that was itself filtered by a lexical proxy; the witness share is lexical overlap, not truth. Phase 1 lost 7 of 247 requests to transport errors and would be inconclusive under the lane's later, stricter rule. We lean on the phase comparison, which shares test sentences and scorer.
- **The ledger case is one run.** One question, one model, one receipt. It shows placement, not model quality; the mechanical blame step also showed a limitation of its own (§6).
- **The tool-use observation is informal.** The runs' tool inventories were not recorded, the configuration was changing that day, and the fabricated-citation report has no preserved transcript (§6.4).
- **Scale and scope.** Everything here is 3B parameters (7B for the ledger and tool-use runs), LoRA or QLoRA, supervised fine-tuning on a few hundred synthetic pairs or about 34,000 weakly labelled ones. We did not try larger models, preference-based objectives, refusal-aware data construction [36], retrieval-trained models [2], or quantization-aware adapters [18, 33]. Any of these might change the training results; none would change the argument of §7 that a lookup the system already holds belongs in the runtime.
- **Same builders, same evaluators.** The systems, datasets, evaluators and this paper come from one team. Nothing here has been replicated by a third party. The evidence files are internal; this paper reports their contents and states where each number comes from.

10 Conclusion

We tried to train grounding into small models and it did not go in. A 3B proposer trained three times to bind protein names and to refuse what it could not bind moved its errors from missed refusals to false ones and back, through a precision mismatch and a lexical cue, and never removed both. A 3B document model learned to produce perfectly shaped records whose relations were rarely right, after learning from labels that mostly were not in the sentence; pushing it to produce more relations produced more from outside the sentence. A 7B model, asked for a protein length, proposed two wrong numbers with confident reasons.

In each case the seam was closed, or held, by something outside the model: a resolver that binds only what the request itself grounds, a witness filter before storage, a ledger that compares a pre-registered number with a receipt. The first of these scored 1.0 on every metric of the same yardstick that three training rounds could not pass, with the first round's weights unchanged, because the task was a lookup and the table was already there. The lesson we take is narrow: before training a model to do something, check whether the system already holds the answer. If it does, the model should not be asked.

References

[1] A. R. Aronson. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In *Proc. AMIA Symposium*, pp. 17–21, 2001. PMID 11825149.

[2] A. Asai, Z. Wu, Y. Wang, A. Sil, H. Hajishirzi. Self-RAG: learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.11511.

[3] F. Brahman, S. Kumar, V. Balachandran, et al. The art of saying no: contextual noncompliance in language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. arXiv:2407.12043.

[4] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer. QLoRA: efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.14314.

[5] B. Dhingra, M. Faruqui, A. Parikh, M.-W. Chang, D. Das, W. Cohen. Handling divergent reference texts when evaluating table-to-text generation. In *Proc. ACL*, pp. 4884–4895, 2019. arXiv:1906.01081.

[6] T. Gao, H. Yen, J. Yu, D. Chen. Enabling large language models to generate text with citations. In *Proc. EMNLP*, pp. 6465–6488, 2023. arXiv:2305.14627.

[7] A. d'Avila Garcez, L. C. Lamb. Neurosymbolic AI: the 3rd wave. *Artificial Intelligence Review* 56(11):12387–12406, 2023. arXiv:2012.05876.

[8] Z. Gekhman, G. Yona, R. Aharoni, et al. Does fine-tuning LLMs on new knowledge encourage hallucinations? In *Proc. EMNLP*, pp. 7765–7784, 2024. arXiv:2405.05904.

[9] S. Geng, M. Josifoski, M. Peyrard, R. West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proc. EMNLP*, pp. 10932–10952, 2023. arXiv:2305.13971.

[10] A. Grattafiori et al. (Llama Team, AI @ Meta). The Llama 3 herd of models. arXiv:2407.21783, 2024. The Llama-3.2-3B-Instruct weights used here are a later release of this family.

[11] E. J. Hu, Y. Shen, P. Wallis, et al. LoRA: low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

[12] B. Hui, J. Yang, Z. Cui, et al. Qwen2.5-Coder technical report. arXiv:2409.12186, 2024.

[13] Z. Ji, N. Lee, R. Frieske, et al. Survey of hallucination in natural language generation. *ACM Computing Surveys* 55(12):1–38, 2023. doi:10.1145/3571730. arXiv:2202.03629.

[14] A. Q. Jiang, A. Sablayrolles, A. Mensch, et al. Mistral 7B. arXiv:2310.06825, 2023.

[15] A. T. Kalai, S. S. Vempala. Calibrated language models must hallucinate. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pp. 160–171, 2024. arXiv:2311.14648.

[16] A. T. Kalai, O. Nachum, S. S. Vempala, E. Zhang. Why language models hallucinate. arXiv:2509.04664, 2025.

[17] P. Lewis, E. Perez, A. Piktus, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2005.11401.

[18] Y. Li, Y. Yu, C. Liang, et al. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.08659.

[19] ggml-org. llama.cpp: LLM inference in C/C++ (including the `convert_hf_to_gguf.py` converter). Software, https://github.com/ggml-org/llama.cpp.

[20] J. Maynez, S. Narayan, B. Bohnet, R. McDonald. On faithfulness and factuality in abstractive summarization. In *Proc. ACL*, pp. 1906–1919, 2020. arXiv:2005.00661.

[21] M. Mintz, S. Bills, R. Snow, D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. ACL-IJCNLP*, pp. 1003–1011, 2009.

[22] Apple machine learning research. MLX: an array framework for Apple silicon, with the `mlx-lm` package used here for LoRA training and fusing. Software, https://github.com/ml-explore/mlx, 2023.

[23] Ollama. Local model server. Software, https://ollama.com.

[24] A. Parikh, X. Wang, S. Gehrmann, et al. ToTTo: a controlled table-to-text generation dataset. In *Proc. EMNLP*, pp. 1173–1186, 2020. arXiv:2004.14373.

[25] S. G. Patil, T. Zhang, X. Wang, J. E. Gonzalez. Gorilla: large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2305.15334.

[26] S. Riedel, L. Yao, A. McCallum. Modeling relations and their mentions without labeled text. In *Proc. ECML PKDD*, LNCS 6323, pp. 148–163, 2010.

[27] P. Röttger, H. R. Kirk, B. Vidgen, G. Attanasio, F. Bianchi, D. Hovy. XSTest: a test suite for identifying exaggerated safety behaviours in large language models. In *Proc. NAACL*, pp. 5377–5400, 2024. arXiv:2308.01263.

[28] O. Sainz, J. A. Campos, I. García-Ferrero, J. Etxaniz, O. Lopez de Lacalle, E. Agirre. NLP evaluation in trouble: on the need to measure LLM data contamination for each benchmark. In *Findings of EMNLP*, pp. 10776–10787, 2023. arXiv:2310.18018.

[29] T. Schick, J. Dwivedi-Yu, R. Dessì, et al. Toolformer: language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2302.04761.

[30] The UniProt Consortium. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research* 51(D1):D523–D531, 2023. doi:10.1093/nar/gkac1052.

[31] B. Wen, J. Yao, S. Feng, et al. Know your limits: a survey of abstention in large language models. *Transactions of the Association for Computational Linguistics* 13:529–556, 2025. arXiv:2407.18418.

[32] B. T. Willard, R. Louf. Efficient guided generation for large language models. arXiv:2307.09702, 2023.

[33] Y. Xu, L. Xie, X. Gu, et al. QA-LoRA: quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.14717.

[34] S. Yao, J. Zhao, D. Yu, et al. ReAct: synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.

[35] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1611.03530.

[36] H. Zhang, S. Diao, Y. Lin, et al. R-Tuning: instructing large language models to say 'I don't know'. In *Proc. NAACL*, pp. 7113–7139, 2024. arXiv:2311.09677.

[37] R. Zhu, Z. Ma, J. Wu, et al. Utilize the flow before stepping into the same river twice: certainty represented knowledge flow for refusal-aware instruction tuning. In *Proc. AAAI Conference on Artificial Intelligence* 39(24):26157–26165, 2025. doi:10.1609/aaai.v39i24.34812. arXiv:2410.06913.

[38] Perslis Research. Fail-first models: failure becomes structure. Research paper, 2026. https://research.perslis.com/fail-first

[39] Perslis Research. Inference placement: where learned inference earns authority in a symbolic system. Preprint, 2026. https://research.perslis.com/inference-placement

[40] Perslis Research. Peel: structural hallucination prevention for offline AAC through symbolic fact authorship. Preprint, 2026. https://research.perslis.com/peel

[41] Perslis Research. The verification floor: scientific integrity that does not degrade with model scale. Pilot preprint, 2026. https://research.perslis.com/verification-floor

A Where each number comes from

Every number in this paper is taken from an evidence file written when the work was done, or from a model-free re-analysis run read-only on 27 September 2026. The paper's claims ledger (`CLAIMS.md`, kept with the paper's sources) maps each number to its file and line; Table 7 gives the classes.

Table 7. Evidence classes behind the paper's numbers.

Result	Source
Proposer rounds, Table 2	each round's evaluator report, written when scored; the evaluator's source (metrics, denominators)
Training settings, Table 1	adapter configurations and training logs for each round; the dataset splits (counted)
Diagnoses of v2; v3 checkpoint	commit messages written at the time; the v3 training log (the failed write)
Resolver behaviour and tests	its source and regression tests; 42 tests re-run from a copy
Table 3	the resolver's grounding step run over both evaluation files, at two versions, with no model
Yardstick audit (§4.5)	set comparisons of prompts across the dataset splits
Document model, Table 4	three evaluation reports (one excluded as truncated); the dataset report; training and export logs; the export script; the manifest of a never-run continuation
Witness audit (§5.2)	the dataset report's kept/dropped counts; the witness proxy's source
Ledger run, Figure 5	the ledger file and head sidecar; chain recomputed; UniProt re-read (P69905, P68871)
Free-form runs, Table 5	seven run logs (events and manifests); configuration history

Reproducing the model-free parts. Three checks need no model and no network except the last.

1. *Ledger integrity.* Recompute SHA-256 over each event's canonical JSON (sorted keys, no whitespace) of sequence number, timestamp, kind, payload and previous hash; it must equal the recorded hash, chain to its predecessor (the first to 64 zeros) and end at the recorded head. All 19 events pass.
2. *Resolver grounding.* For every prompt of both sets, compare the identifiers the request grounds (§4) with the gold label, then feed a wrong-identifier claim; all 62 pass at both resolver versions.
3. *UniProt lengths.* Read the canonical entries for P69905 and P68871 (142 and 147 on 27 September 2026).