fineSample

The Base GAR

Answering from what an organization means. A first look at grounded answer reasoning: our research into a small model that answers everyday questions from an organization's own definitions and records, or says what is missing.

Compile what is known, learn what remains.

Research note, November 2026

This is early research. We have been working on it for about five weeks, on and off. Every result below comes from development data, and none has been scored on a fresh test set yet. We are publishing the idea and the method now, and we will publish the first fresh result when it is in, whichever way it goes.

The organizations, customers and records in the examples are made up.

About this note

Where our research into grounded answer reasoning stands, and how we will test it.

A research note from the fineSample lab, November 2026. Every result in it is development evidence, and the full working paper follows the first fresh test.

In one paragraph

Base is our research into a compact model that answers everyday questions about an organization's own records. It reads each question against definitions the organization's experts write down in Braid, a language we created, and answers from the records with the evidence, or abstains and says what is missing. No pretrained or hosted language model runs inside it. Early results on development data hold up in places and fall short of our bar in others, and no fresh test has been scored yet.

1	What grounded answer reasoning means	4
2	An inversion	6
3	How Base is put together	7
4	Where it sits among today's options	13
5	What we have measured so far	15
6	What hasn't worked yet	17
7	How we will judge it	18
	Scope and limits	19
	What's next	20
	Sources	20

Introduction

Most tools either answer only what was built in advance, or guess.

People in any organization ask the same kinds of question all day. Which invoices from Acme are still open? How many deliveries left the north warehouse last week? Who has the loaned projector? The wording changes from person to person and from month to month. The meaning doesn’t. The organization decided what counts as an open invoice or a delivery, and the answer is sitting in its own records.

Most tools answer these questions in one of two ways. Either someone built the answer in advance, as a dashboard, a defined metric or a chatbot intent, or a large hosted model reads the question and the database layout and writes a query. The first way is reliable but only covers what was built. The second covers almost anything, but when it is wrong it returns a table that looks exactly like a right one.

The Base GAR is our research into a third way. This note covers what we mean by grounded answer reasoning, how Base is put together, what we have measured so far and what we haven’t.

An answer with its records, a prediction, or a plain statement of what is missing.

We use **grounded answer reasoning**, or GAR, as the name for a kind of model. Each word does a job.

- **Grounded:** every answer is tied to meaning the organization wrote down and to evidence from its own records.
- **Answer:** the output is an answer to the request, not a label or a route to some other system.
- **Reasoning:** the model applies the organization's meaning to the evidence to reach the answer.

A GAR model gives one of three outputs:

1 an answer, with the records it was computed from;

2 a prediction for a stated target and time horizon, with its uncertainty;

3 an abstention that names what is missing.

The third one matters most. A wrong answer that looks right costs more than a refusal that says what it would need. GAR is a name for what the model is for. It isn't a claim about how the model works inside, and it isn't a claim that it works better than anything else.

Put as needs, a model like this has to:

1 read a request the way the organization means it;

2 answer from the organization's data and show the evidence;

3 abstain when the reading isn't settled;

4 get better from confirmed use, so that confirming “deliveries means shipments” once helps with delivery questions it was never taught;

5 be useful on day one, before anyone has written down everything the organization means.

The scope is a department or a whole small business: finance, supply chain, operations and customers. It is not a general assistant, and it does not read free-form documents.

Start from what the organization means, and learn only the words.

Our thesis says we don’t build models, and Base is a model. The line was about general-purpose models, the kind that get bigger, better and easier to swap every year. Base is built the other way round.

A general model learns a great deal of language first and then guesses what a business term means from its training and the prompt. Base starts from the meaning. The organization writes down what its terms mean, that meaning is compiled into the model’s structure, and learning fills in only what is left over: mostly how people word things. The design rule fits in one line:

> Compile what is known, bind it to the world, learn only what remains uncertain.

Our thesis starts from the idea that a fine sample is enough: you don’t need to see everything to act well, if what you have is the right evidence. We think an organization’s written meaning can work the same way. It shouldn’t have to be complete. The bet is that it is a good enough sample of what the organization means to start answering from, with learning filling in the blanks through use. We haven’t measured yet how incomplete it can be before that stops holding.

That is what we mean by Base being native to the organization it serves. By design, the part that lasts is the organization’s own definitions, and the learned part is small and kept apart from them, so retraining it shouldn’t change what any term means, even if it changes which reading gets picked. It also keeps the rest of the thesis: an answer should point to its record, and “missing” and “not yet” are honest results that shouldn’t be rounded up.

Written definitions, connections to the records, and a small learned layer.

Base has three parts.

The organization's definitions. These are written in Braid, a structured language we created for Base that reads like plain English. It has a fixed grammar, and the compiler rejects anything outside it. Here is an excerpt from one of our test organizations, a made-up wholesale distributor:

```
Under Sales:
  A Customer is a business or person that buys from us.
  A Customer is identified by customer id.
  A Customer is also called client, account.
  A Customer has:
    customer id: identifier, required, unique
    segment: retail, trade, or wholesale, required
    credit limit: money, required
    in arrears: yes/no/unknown, optional
  A Customer is in arrears when:
    it has a Finance.Customer Invoice through customer where:
      status is issued
      and due on is before today
```

These definitions compile into the model's structure. They decide which readings of a question are possible at all, and learning can't redefine them.

Braid rests on a few plain rules:

- **Meaning and logic share one source.** Rules can only use terms the definitions contain. Order in the file doesn't matter, since names resolve across the whole module.
- **An undefined term doesn't compile.** If a rule uses a term nobody defined, the compiler doesn't guess the closest match. It asks for a definition. A term that is almost defined is still not defined.
- **Whose word it is matters.** Finance and Risk can both define "Exposure" and mean different things. A rule has to say which one it means, or it won't compile.
- **Missing is not zero.** A missing amount is unknown, not small. It stays unknown through every step that uses it.
- **Where data lives is kept separate.** If a database column is renamed, only the connection changes. The definitions and the answers stay the same.
- **Learning stays inside limits.** On its own, learning can't add a concept, a rule or an outcome. New aliases are added only within limits the organization sets, and anything more goes back through its review. Every change is logged and can be rolled back.

Connections to the records. A separate file says where each concept lives, for example that customers come from the `customers` table and that credit limits are stored in cents. Answers are computed from these records, not written as prose.

A small learned layer. No pretrained or rented language model runs anywhere inside Base. Today the reader uses word-to-part tables fitted from confirmed questions. They start from the names and aliases in the definitions, not from a pretrained model, and confirmed use moves them from there. Alongside that, we trained an English encoder from random starting values to take a larger part in reading. It has about 7.3 million parameters, and it isn't connected to the reader yet (more on that below). A typical small open model that a company might host itself has 3.8 billion, roughly 520 times more. The small footprint is a constraint we chose, not a ceiling, and we can add parameters if the end goal calls for it. The size says something about footprint and provenance, not about quality.

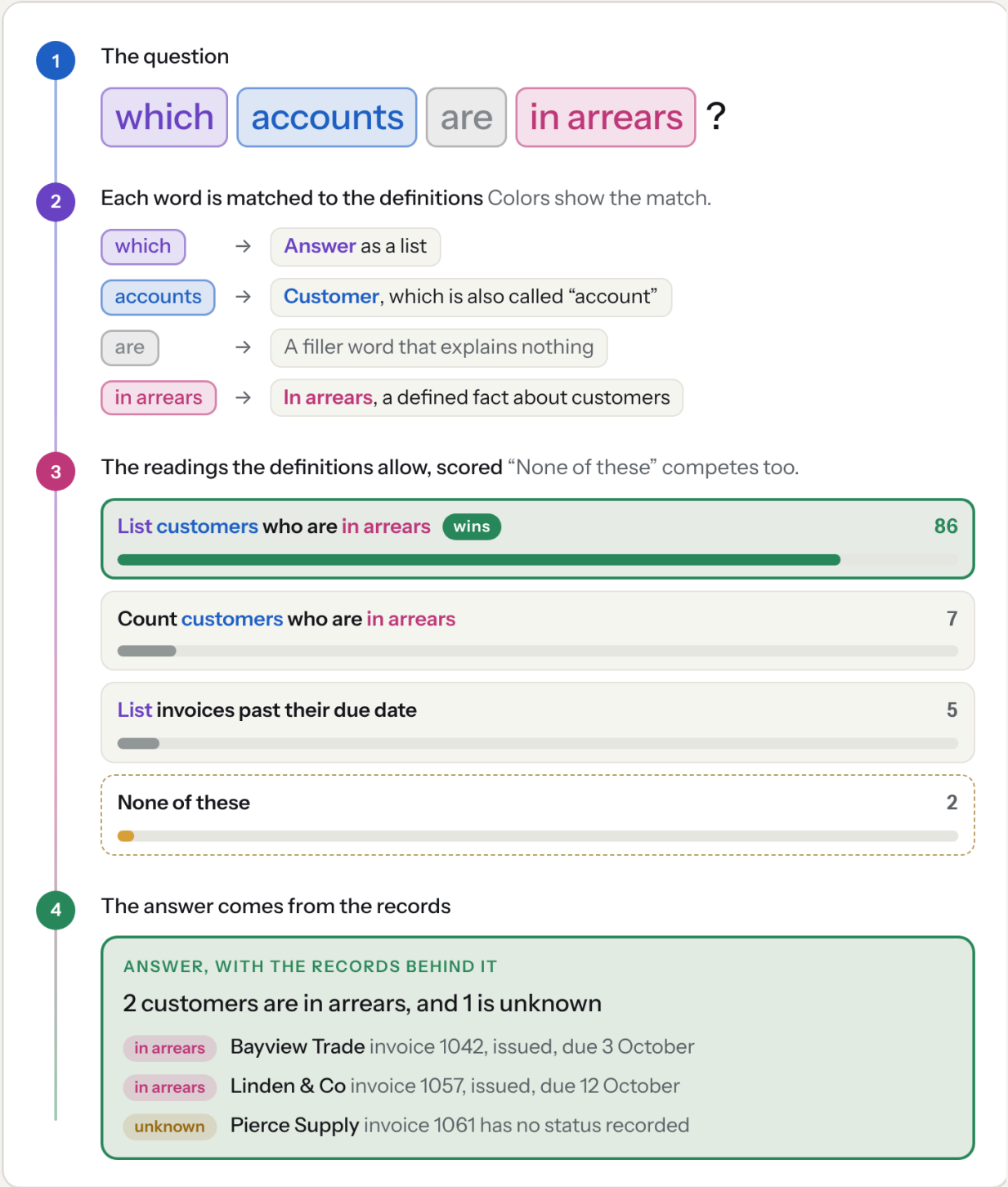


Figure 1. How Base reads a question, shown for the design. The definitions decide which readings are possible. The reader picks the reading that best explains the question's words, and in the design “none of these” competes on equal terms with every reading. The chosen reading runs against the records, and the result is an answer, a prediction or an abstention. The scores and records are made up to show the mechanism. On the web page the figure is interactive.

Reading a question

Base reads a question by asking which of the readings the definitions allow would best explain the words that were used. In the design, “none of these” is scored the same way as every reading, so abstaining is one of the answers the model can give rather than a filter added afterwards. Today’s reader still abstains through separate checks: a test that every word in the question is explained, and a margin against close rival readings. The next design is meant to replace them.

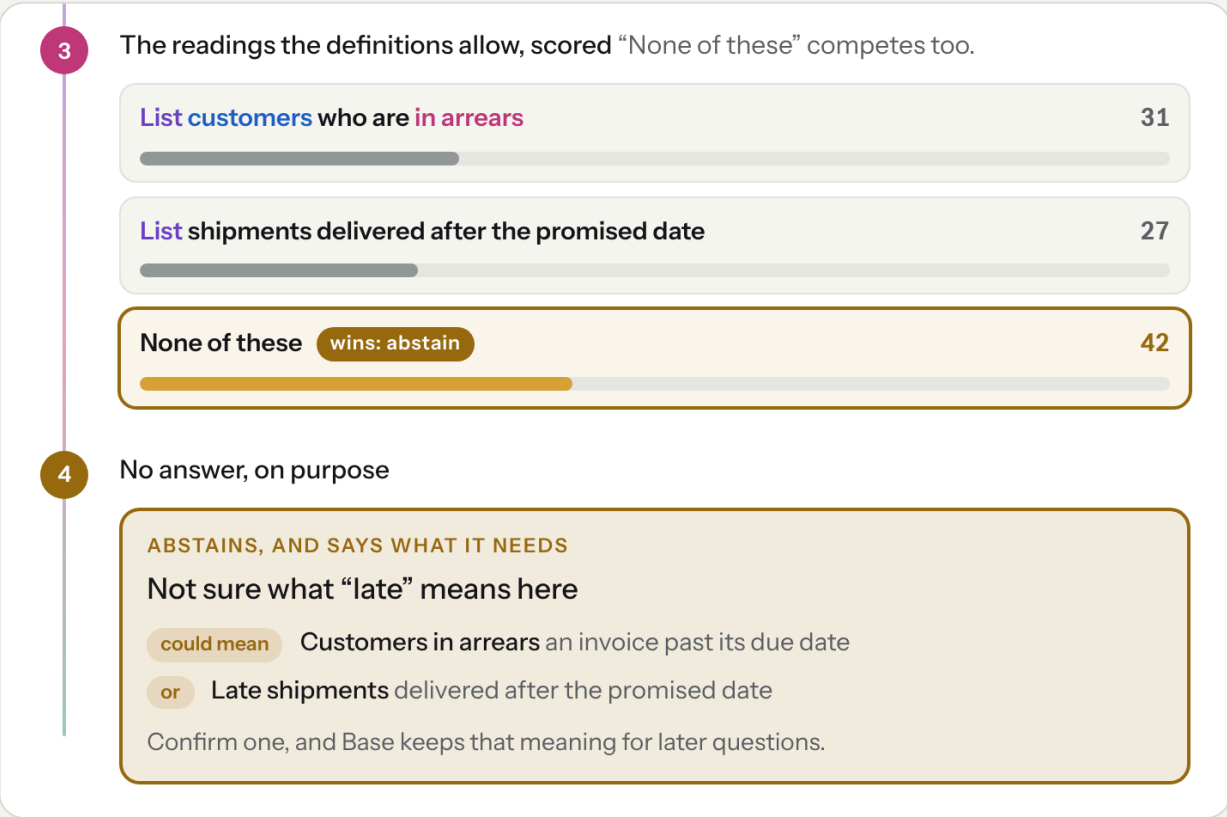


Figure 1, continued. For “Which accounts are late?” no definition explains “late”, so “none of these” wins and Base abstains, naming the two meanings it could take.

Keeping unknowns visible

Records are often incomplete. Base runs every reading with three truth values: true, false and unknown. That way a partial answer can’t pass for a complete one.

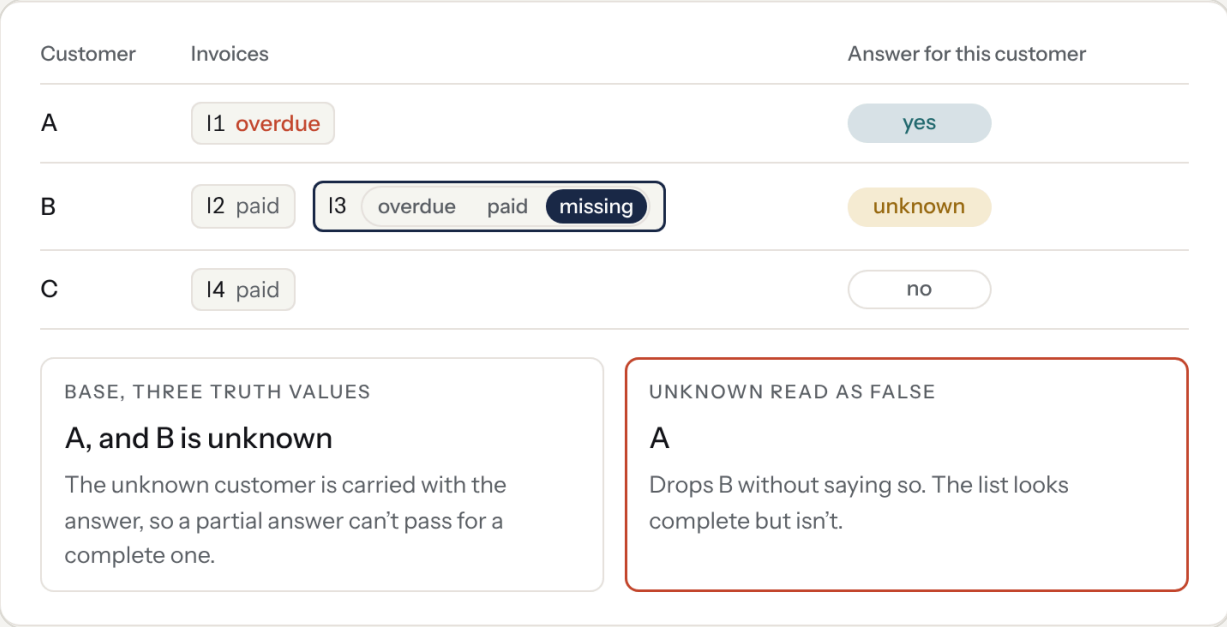


Figure 2. "Which customers have an overdue invoice?" over three customers. A has one, C has none, and B has an invoice with no status recorded. Base answers "A, and B is unknown". A system that treats unknown as false would say "A" and quietly drop B. Asked the opposite question, "which customers have no overdue invoice?", Base answers "C, and B is unknown", where the same system would claim B. The records here are made up for illustration.

Learning from use

When a person confirms a reading, that confirmation becomes teaching. What proves useful over time is meant to go back into the definitions as new aliases, relationships or refined rules, under the organization's review, so knowledge lives where people can read it rather than only in weights. This part is designed and hasn't been run on the current reader yet.

Who writes the definitions

The definitions are written by domain experts inside the organization: the people who already know what an open invoice or a late delivery means there. Nobody wants to write a complete account of their business before asking a first question, and they don't have to. Because Braid reads like plain English, experts can have an AI assistant draft definitions and then review and correct them. The assistant is only used while the definitions are written. It is not part of the running model, and no question or record passes through it when Base answers. How much review a drafted set of definitions needs is one of our open questions.

What Base doesn’t do

- It doesn’t send questions or data to a hosted model.
- It doesn’t write free-form database queries. It builds a reading the definitions allow and runs it.
- It doesn’t let learning change what a term means.

Eight ways organizations answer these questions today, and where Base differs.

Organizations already answer these questions in eight broad ways. Base is a ninth design, and we haven't measured it against any of them yet.

Approach	How it answers	Where it struggles
Dashboards and SQL reports	Analysts build fixed views	Questions nobody built a view for
Semantic or metrics layers	Metrics defined once, compiled to SQL	Anything not defined, and exact names only
AI copilots that write SQL	A hosted model writes a query from the question and schema	Plausible wrong tables, data leaving the building
Chat over documents	Retrieve passages, then a model writes an answer	Wrong or unsupported answers, counts and totals
Intent chatbots	Classify into a fixed list of intents	Anything outside the list
Self-hosted open models	A pretrained model tuned and run locally	No built-in abstention, weights nobody inside wrote
Knowledge graph platforms	Entities and links modelled by hand	High modelling cost
Human analysts	A person finds and checks the answer	Queues, turnaround, knowledge that leaves with people

The published numbers for AI copilots are worth knowing, with their dates, since systems improve quickly. On BIRD, a public benchmark of questions over databases, GPT-4 reached 54.9% in 2023 against 93% for data engineers and students; current systems score much higher. On Spider 2.0's enterprise tasks, published in 2025, a leading agent solved 21%. On

BEAVER, built from private company warehouses, agent frameworks reached about 11% in 2026. Retrieval doesn't remove the problem either: a 2024 Stanford study of three commercial legal research tools found each gave wrong or unsupported answers on 17 to 33% of queries.

None of that tells us Base will do better. What we can say is where the designs differ:

- **Where meaning comes from.** In copilots and chat assistants a business term means whatever the model and prompt make of it. In Base it is written by the organization. That puts Base close to semantic layers and knowledge graphs, with the difference that Base learns everyday wording itself.
- **Where it runs.** Base is small and local, with no hosted call.
- **How wrong answers are avoided.** Unknowns are reported rather than dropped, and in the design abstaining is a reading the model can choose.
- **Who writes the meaning.** Domain experts write it in a language that reads like plain English, with an AI assistant drafting if they like, rather than engineers encoding it by hand.

One confirmed word carried into questions it was never taught.

Everything in this section is development evidence. These are results on data we have already looked at, mostly written by us or drafted for us, across three small made-up organizations: a wholesale distributor, a group of animal clinics and a field service business.

A word confirmed once carried into questions with other meanings. On the wholesale distributor, 29 right and 6 wrong out of 240, with the rest abstained. With the settings frozen and moved to the animal clinics, 86 right and 12 wrong out of 222, where before each confirmation it abstained on all 222. Both used an earlier version of the reader.

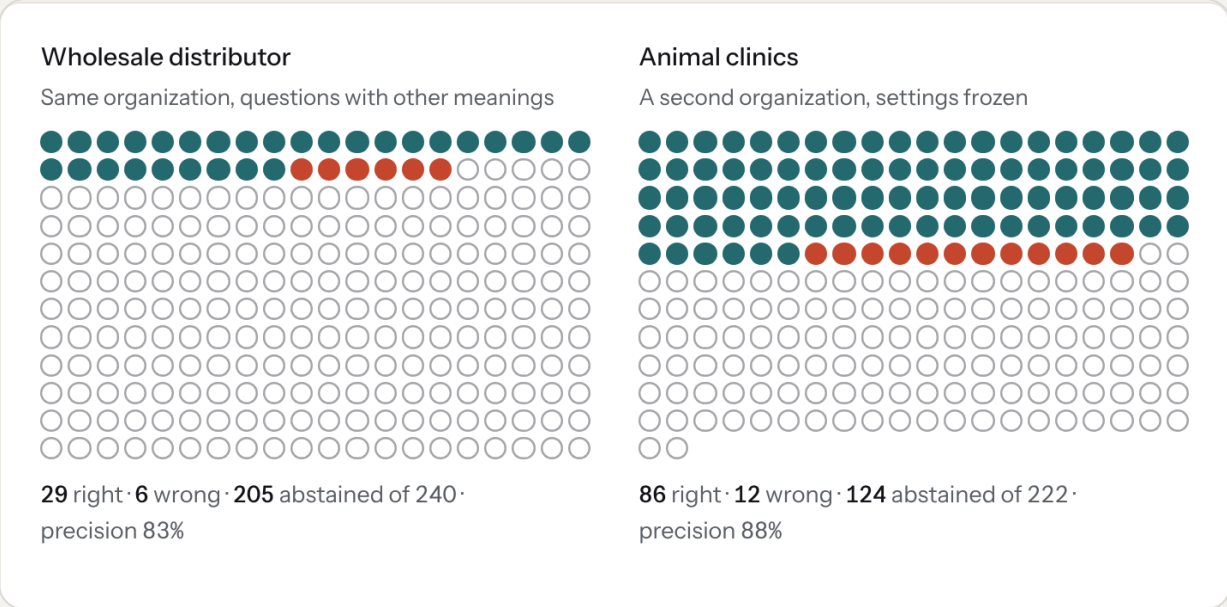


Figure 3. Each dot is one test question. “Abstained” covers every question the reader didn’t commit to. Before each confirmation, all of these questions were abstained.

An English encoder learned from scratch. Our 7.3-million-parameter encoder, trained on 160 million tokens of licensed English text, beat a same-size control trained on shuffled text on every criterion we set in advance, including 335 against 233 of 395 grammar test pairs. On predicting

hidden words it scored 44.7% against the control's 24.8%, but each was scored on its own held-out text, ordered for the encoder and shuffled for the control. That gap shows how much word order helps, not that one model beat the other on the same test.

Answers from connected records were exact. On a public procurement dataset, matching invoices to orders and receipts agreed with the reference on all 5,671 cases, at a quarter of a millisecond for 95% of them. A separate check answered 4,200 templated questions over connected records with nothing wrong, on an earlier system with hand-written question forms that hasn't been merged into the current line. These test the connection to records, not the reading of real questions.

The encoder isn’t connected yet, and the search often misses the right reading.

We are putting this on the record next to the results.

- **Connecting the encoder to the reader failed.** The encoder, trained from scratch, passed its own English tests. When we matched its states for the question’s words against its states for the definitions, it could tell what kind of thing a word referred to, but not which specific part of the organization’s meaning it meant. The definitions added almost nothing to that match, and the scoring around it had defects that made confident wrong answers easy. It failed all eight development criteria. The reader runs today without the encoder, but it has not yet reached the accuracy bar.
- **Proposing the right reading is the main limit.** For most of our development questions, the right reading never makes it onto the list of candidates the reader chooses from. Most of that loss is in the search, and the rest is in forms the candidate builder can’t build yet.
- **No learning curve yet.** Learning from use is designed but hasn’t been run on the current reader.
- **A separate confidence check didn’t work.** We tried judging confidence with a separate component and dropped it. Abstention now has to come from the one model.

Three lessons from these failures shape what comes next:

1 **Having the right reading isn’t the same as being able to commit to it.** A reader can hold the right answer among its candidates and still have no evidence that lets it say so.

2 **Gains on synthetic wording didn’t carry to how people actually write.**

3 **Learning a new part can break what already worked.** Each change now has to show it keeps what passed before.

One fresh test, frozen in advance, published whichever way it goes.

Every test is written down and frozen before it runs, and any departure from a freeze is recorded beside the result. Some development bars were set after the numbers were seen, and we mark those. Development tests can stop a line of work but can never approve one.

Two fresh test sets are recorded and set aside. No score has been run on either set, and nobody working on the reader has read them. To pass, at least 80% of the readings Base commits to on fresh questions must have exactly the right meaning, with the lower end of a 95% interval at 65% or above. Abstentions are not counted as wrong. A pass would be one fresh result, not a release. Over time the goal is 90%. When the first fresh test is scored, we will publish the result and the full working paper, whatever the number is.

Scope and limits

What this note does not establish.

- The evidence comes from three small organizations, and most of the wording was written by us or drafted for us.
- How much review an assistant-drafted set of definitions needs hasn’t been measured.
- Base is for questions about an organization’s own structured records. It isn’t for open-domain questions or for reading free text.
- The counts are small, and we report them as counts.

What's next

Better candidate readings, then learning from use, then the fresh test.

The next work is on proposing better candidate readings, then learning from use with the teaching deleted afterwards to check that gains stick, then the first fresh test. The idea stands as a design today, not a result. We think it is worth showing early because the question it asks is a useful one: how far you can get when the meaning comes from the organization and the model only has to learn the words.

Sources

Sources.

- Li et al., 2023, BIRD
- Lei et al., 2025, Spider 2.0
- Chen et al., 2026, BEAVER
- Magesh et al., 2024, Hallucination-free? Assessing the reliability of leading AI legal research tools
- Abdin et al., 2024, Phi-3 technical report
- dbt MetricFlow documentation
- Rasa fallback classifier documentation
- Palantir Foundry Ontology documentation
- BPI Challenge 2019, the public procurement dataset used for the record checks