A metadata table answered 45 of 56 aggregate questions where reading the documents answered 30 — and used 21.6 times fewer tokens

By the Deasy Labs team at Collibra

Executive summary

Enterprise questions are often aggregate: how many suppliers are in this region, which product line has the most incidents, what is the median age of these contracts. Retrieval finds the passage that answers a question, and an aggregate question has no such passage. The answer is spread across the corpus.

We tested whether extracting a few attributes into a queryable table lets an AI agent compute those answers instead of reading for them. Deasy supplied the metadata, tagged from the same documents the agent would otherwise have had to read. Same agent, same model, same budget throughout; the only variable was which tools it had.

On a corpus of 483 S&P 500 company articles, an agent with a metadata table answered 45 of 56 questions exactly correctly. The same agent reading the documents answered 30 of 56, and used 21.6 times more tokens doing it.

On a corpus of 500 Nobel laureate articles, the two approaches tied: 42 of 56 either way. That domain's result is a cost result only, at 7.0 times fewer tokens.

The mechanism is the part worth remembering. Answering from a table costs about the same whether the answer touches one document or five hundred, while reading costs more with every document involved. On the organisations corpus the saving ran from 8 times on the narrowest questions up to 48 times at 21 to 60 documents, then fell back on the broadest.

Measured against independently verified answers, the values Deasy extracted agreed 98.6% to 100% of the time, field by field. That is the direct measure of how good the metadata was, and it is separate from how many questions the agent then got right.

Every figure here is a query-time cost. What the table cost to build is not measured.

What this covers, and what it does not

Two corpora, 56 questions each, one model, one agent framework. Nothing here generalises beyond that footprint without further work. What is measured is the query-time value of a metadata table, given metadata of a quality stated field by field below. What is not measured: what any product delivers on installation, and what the table cost to build. Threats to validity, near the end, is the full list.

The problem: aggregate questions have no passage to retrieve

Document retrieval works by finding the piece of text that contains the answer. That is the right design for "what is the termination clause in the Acme contract" and the wrong design for "how many of our contracts auto-renew".

The second question has no answer anywhere in the corpus. It is a property of the whole collection, so producing it from documents means visiting enough of them, extracting the same field from each, and computing. An agent can do that, by searching, opening files, and keeping a tally in its context window. The cost scales with the number of documents involved, and so does the number of chances it has to miss one.

The hypothesis tested here is that if the corpus has already been tagged into a table, the agent can compute instead of read: more accurate and much cheaper at once, with the advantage widening as more documents are involved.

Terminology

Term	What it means here
Aggregate question	A question whose answer is a property of a set of documents rather than of any one of them: a count, a ranking, an average, a median, or a complete list.
Exact-answer rate	The share of questions answered exactly correctly. A numeric answer must match; a list answer must contain precisely the right set of names, no more and no less. There is no partial credit.
How many documents a question's answer depends on, its breadth	The number of documents that contribute to the correct answer. A question about companies founded before 2000 depends on 424 documents; a question about a state with a single company headquarters depends on one.
A question whose true answer is none	A question with no matching entities, included to test whether a system invents results.
Tokens per question	Every token the model read and wrote, including repeated reads of cached context. It measures work done, so it compares fairly across configurations that work in very different ways.
Verified reference answers	The correct answers, computed by a separate deterministic parser reading the same articles the agent sees. Never taken from the tagging system under test.

How the evaluation was built

The agent, and the four configurations

The agent is Claude Sonnet driven by Claude Code as an autonomous loop: for each question it plans, calls tools, observes results, and iterates until it commits to an answer. Every configuration

used the identical agent, model and spending limit. The only variable was which tools it had and what data those tools reached.

Configuration	SQL over the metadata table	Search and read the documents	Why it is in the study
Gold-table ceiling	yes	no	What a perfect table achieves
Metadata table	yes	no	The configuration under test
Document reading	no	yes	The read-the-documents baseline
No information	no	no	What the model already knows

The two middle rows are the comparison. The two outer rows are what make it interpretable. And the two table-backed rows differ only in what fills the table: the one under test holds tags Deasy extracted from the corpus, while the ceiling above it holds the same columns filled from the verified answers instead. That pairing is what turns the comparison into a measure of the metadata rather than of the idea.

The gold-table ceiling is a table built from the verified reference answers rather than from extracted tags. It scored 98.2% and 100%, which tells you the questions are answerable, the strict scoring is fair, and the agent knows how to use a table. Any shortfall in the configuration under test is therefore attributable to the metadata. Without this row, a score of 80.4% could equally mean "the table approach is wrong" or "the table has gaps", and those need different fixes.

The no-information floor is the same agent with no corpus and no table. It scored 17.9% and 26.8%, and most of what it gets is the cheap kind: on the organisations corpus, 4 of its 10 correct answers are questions whose true answer is none, which a tool-less model collects by guessing zero. That leaves 6 of 56 that look like genuine recall, and 8 of 56 on the people corpus. A floor of exactly zero would be more suspicious than this one.

Document reading was given shell commands, file search, glob patterns and file reads over the corpus, plus a $2.00 per-question ceiling that nothing came close to exhausting: the most expensive single question cost $1.79, and single questions ran to 112 turns and 78 tool calls. It was not throttled. It is also not a vector-search pipeline: there is no embedding index and no reranker, because an index's own retrieval quality would become a second variable. What it models is an agent searching and reading, the common unaided case, not a tuned retrieval stack.

The corpora, and where the correct answers came from

Two deliberately unrelated domains, so a positive result is not a single-domain artifact.

Organisations. 483 S&P 500 companies. Attributes: headquarters state, or country for the non-US constituents, and founding year.

People. 500 Nobel laureates across all six prizes. Attributes: birth country, birth year, and country of death, which is correctly null for the living.

Both were built the same way. Entities were seeded from Wikidata, then the English Wikipedia article for each was fetched and kept as the document corpus. Those articles, and nothing else, are what document reading sees.

Deasy was given exactly the same articles, rendered as PDFs because that is its expected input, and tagged them against a defined schema: headquarters location and founding year for the companies, birth country, birth year and country of death for the laureates. Those tags, joined back by document, are the entire contents of the table the agent queries. Nothing else was added to it, and no fallback to the documents was available when a value was missing. How accurate the tags turned out to be is measured below, and it is the number that describes the metadata.

The correct answers came from neither. They were extracted from the same articles by a separate deterministic parser, never from the tags and never from Wikidata. That distinction is load-bearing: the seed database decides only which entities exist, not what is true about them. If the tags were also the answer key, the study would measure self-consistency and nothing else, and a reading agent that read a document correctly would be marked wrong wherever Deasy disagreed.

Entities whose attribute could not be resolved unambiguously from the article were dropped from the corpus rather than guessed. A company whose article names two headquarters is not in the corpus at all, so no configuration is penalised for reading it correctly.

Questions were generated programmatically across 11 families of aggregate operator, listed in the appendix, 56 per domain, spanning from questions with no matching documents at all to questions that depend on every document in the corpus. Every reference answer is recomputed by an audit that also checks the table, the questions and the scorer, and both corpora pass with zero errors.

One consequence bounds the whole result: because the questions are computed from the same attributes the table holds, every question is answerable from the table's columns by construction. The share that asks about something nobody extracted is zero here and will not be zero anywhere else.

Scoring

Scoring is deterministic. There is no model acting as a judge.

Each question family has a strict rule: exact match for counts, distinct counts and medians; a tolerance of half a year for averages, which permits a rounding difference but not a wrong subset; name and value both correct for earliest and latest; and set equality for list-every-member questions, insensitive to order. A language model as grader would have introduced exactly the kind of unquantified noise this study set out to measure, and would have destroyed the most useful signal in a wrong answer, which is how far off it was.

Set equality is unforgiving. One missing or one extra name in a list of sixty fails the entire question. That strictness is deliberate, but note which way it biases the result: it penalises small residual gaps

in the metadata heavily, and therefore probably understates how useful the table would be in practice.

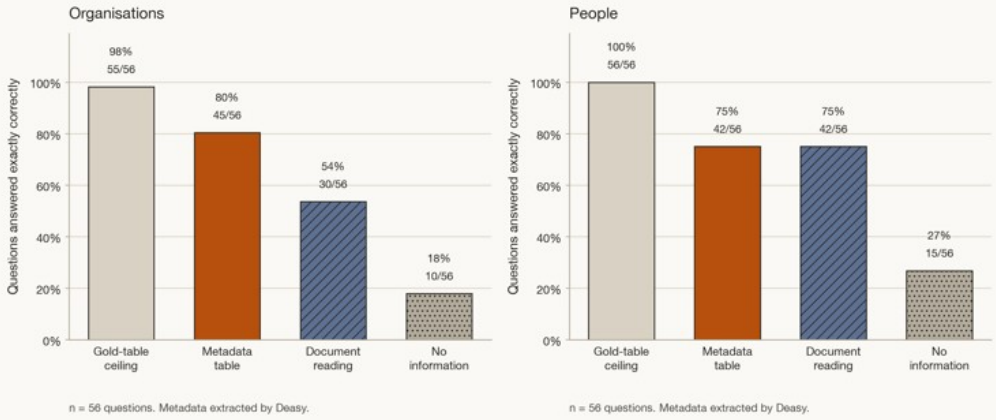
An objection worth answering first

Both corpora are public encyclopedia content, well represented in the model's pre-training, so the first question a sceptic asks is whether the model is simply reciting.

Two things address it. The no-information floor measures the recitable fraction directly, and once the guessable zero-answer questions are set aside it is 6 and 8 questions out of 56. And every question is scoped to the pinned collection, phrased as "among the companies covered in this corpus", so the answer depends on the exact 483 or 500 entities present, which world knowledge does not supply. This reduces the concern substantially rather than eliminating it.

Results

Accuracy



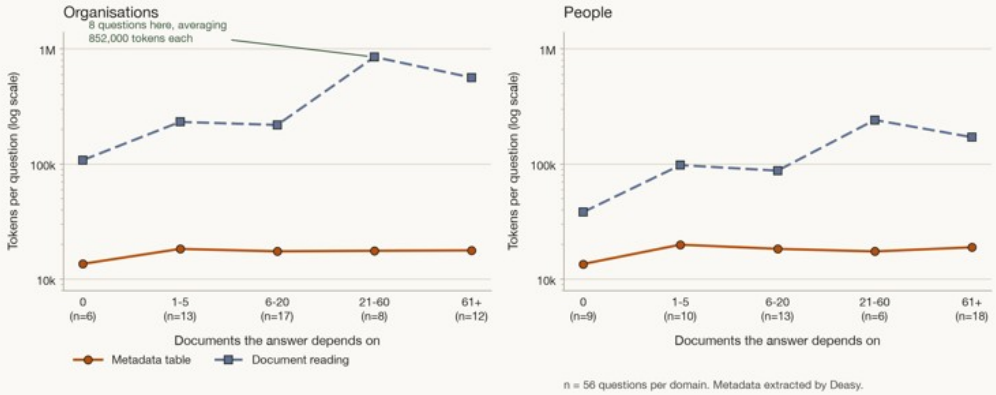
A table built from the verified answers shows the questions are answerable and the grader is fair, and a model answering from memory alone shows they are not guessable. Between those bounds, the metadata table answers 15 more questions than document reading on the organisations corpus and exactly as many on the people corpus. n = 56 questions per domain.

On the organisations corpus, the metadata-table configuration answered 45 of 56 questions exactly correctly, against 30 of 56 for document reading. That is a margin of 15 questions, or 50 percent more questions answered.

On the people corpus, both configurations answered 42 of 56. That is a tie, not a narrow win in either direction, and it should not be read as one. What that corpus produced is a cost result at identical accuracy.

The gold-table ceiling reached 55 of 56 and 56 of 56. Its one miss is itself a tie: two California companies share a founding year of 2015, the question asks which was founded most recently, and the answer key holds both names while the agent gave one.

Cost, and why it scales the way it does

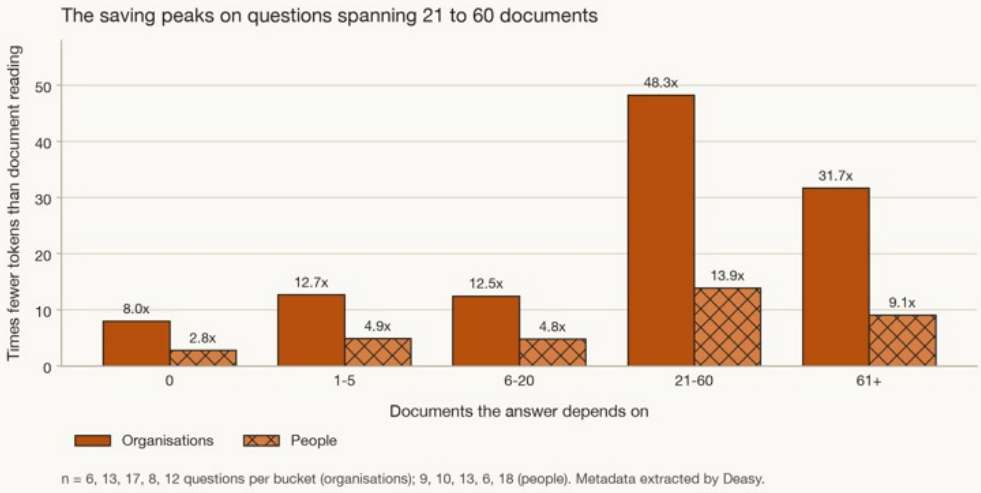


Answering from a table costs about the same whether the answer spans one document or five hundred, while reading the documents costs several times more and rises steeply on the broader questions: a flat cost against a growing one. Tokens per question, log scale. n = 56 questions per domain.

The metadata-table configuration used between 13,545 and 19,938 tokens per question across every breadth bucket in both domains, whether the answer touched one document or five hundred. A SQL aggregate costs the same regardless of how many rows it spans, and this table is small enough that a schema, a query and a result fit in a fixed amount of context. That last clause is a condition, not a detail: the flat line depends on the table staying small enough to query without first searching for the right columns.

Document reading is the opposite. On the organisations corpus its bucket averages climb from about 108,000 tokens per question on questions with no matching documents to about 852,000 on questions spanning 21 to 60. Individual questions go much higher: the most expensive single question in that configuration consumed 2,363,471 tokens. The mechanism is not subtle. To aggregate over N documents by reading, the agent has to get N documents' worth of text through its context window, and it does so imprecisely, with false positives from searching, repeated reads, and re-derivations after it loses track.

Overall, the table used 21.6 times fewer tokens on the organisations corpus and 7.0 times fewer on the people corpus.



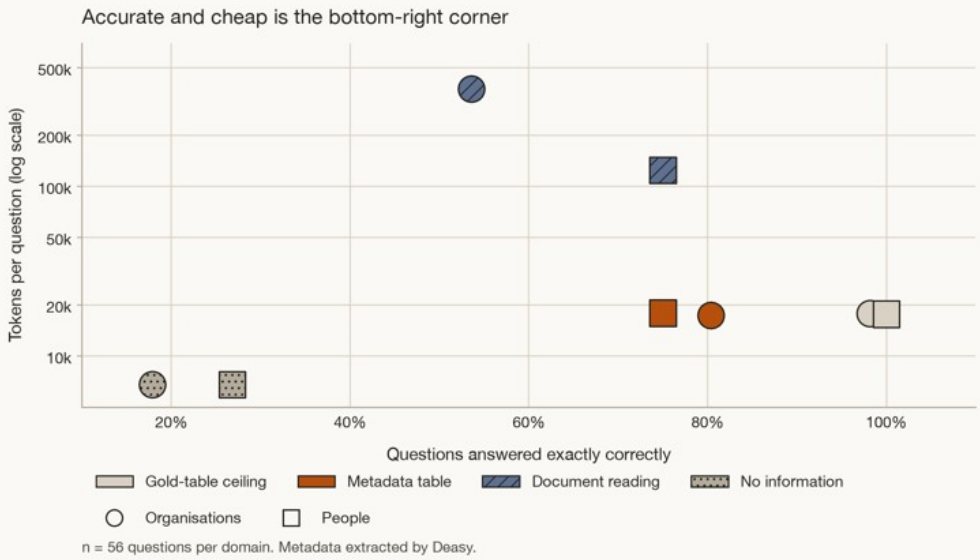
The cost advantage of the table climbs from roughly threefold to eightfold on the narrowest questions to between 14 and 48 times on questions spanning 21 to 60 documents, then falls back on the very broadest. The saving peaks with the breadth of the question rather than growing without limit.

The ratio is not constant, and its shape is what lets you predict what would happen on your own corpus. On questions whose answer is none, the table saved 8.0 times on the organisations corpus and 2.8 times on the people corpus, because even proving an absence requires the reading agent to search. The saving then climbs to 48.3 times on questions spanning 21 to 60 documents.

Then it falls back. On the widest questions the multiple drops to 31.7 times on organisations and 9.1 times on people. That contradicts the tidy story, so it is worth stating plainly: the advantage peaks rather than growing without limit. In the traces, several of the widest questions are ones where the reading agent stopped searching early and answered anyway, which lowers its cost without improving its answer, though that is a qualitative reading rather than a measured quantity.

At list prices for this model, the organisations corpus cost about one cent per question with the table and 33 cents by reading. Treat that as order of magnitude only: dollar figures depend on the model, the provider and the price list, and they shift with caching effects that depend on the order runs happen to execute in. The token ratios survive a price change; the dollar figures do not.

Accuracy and cost together



The table-backed configurations occupy the cheap and accurate corner in both domains, while document reading sits an order of magnitude higher in cost with no matching gain in accuracy. Accuracy increases rightward and cost increases upward, so lower-right is better. Tokens per question, log scale. n = 56 questions per domain.

Accuracy and cost normally trade off, because more accuracy means reading more. The point of putting both on one frame is that here they do not. On the organisations corpus the table is both more accurate and 21.6 times cheaper. On the people corpus it is equally accurate and 7.0 times cheaper. In neither domain did the extra 107,000 to 357,000 tokens per question that document reading spent buy any accuracy at all.

One question, both ways

The question: *Among the S&P 500 companies covered in this corpus, how many are headquartered in the U.S. state of Texas?* The correct answer, computed independently from the articles, is 49. It depends on 49 documents.

With the table, the agent issued one statement:

```
SELECT COUNT(*) FROM organisation WHERE headquarters_state = 'Texas'
```

It got back 49, submitted it, and stopped. Four turns, two tool calls, 17,601 tokens.

Reading the documents, the agent listed the corpus, then searched for “Texas” and got 186 files back, because Texas appears in a great many articles that are not headquartered there. It wrote progressively more elaborate patterns, enumerating Texan cities by hand: Houston, Dallas, Austin, San Antonio, Fort Worth, Plano, Irving, Midland, The Woodlands, Frisco. It searched for “headquartered in” near each of them, searched for “S&P 500 component” to work out which files were even in scope, and opened 51 articles to adjudicate what the searches could not settle. In

total: one corpus listing, 19 searches, 51 file reads, 7 shell commands, 112 turns, 1,338,440 tokens, about seven and a half minutes.

It then answered with a narrative paragraph, a list of 38 company names, and the number **38**. Eleven companies short.

The table configuration used 76 times fewer tokens on this one question, and it was the one that was right.

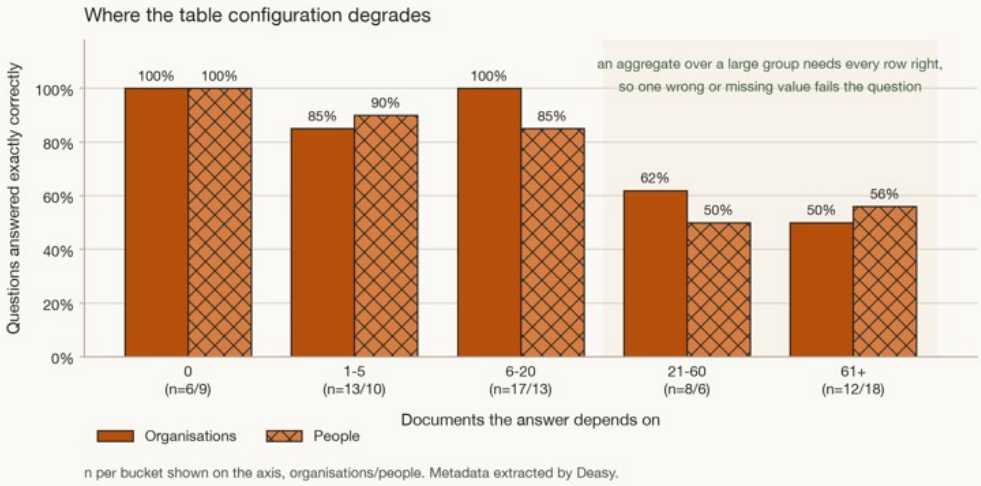
Nothing about the reading agent's behaviour was unreasonable. Its searches were sensible, its city list was good, and it audited the ambiguous cases by hand. What it could not do was stop. Recovering a field from prose means disambiguating every document individually, and there is no point at which an agent working question by question knows it has finished.

The table had that work done once, in advance. So, for that matter, did the deterministic parser that produced this study's answer key, at negligible cost. The transferable lesson is about doing the extraction once rather than once per question. A tagging product is one way to do that and not the only one.

Where this breaks

Two limits, in order of how much they should change your mind.

The table configuration degrades on the broadest questions



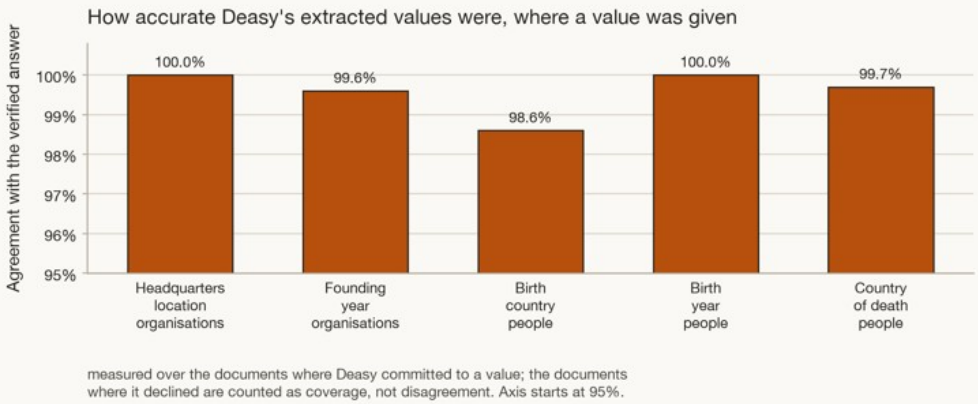
The metadata-table configuration answers 85 to 100 percent of questions spanning up to twenty documents and 50 to 62 percent of those spanning more, because an aggregate over a large group depends on every row being right and a single wrong or missing value moves the answer off an exact match. n per bucket is shown on the axis, organisations then people.

Up to twenty documents, the table configuration answered between 85% and 100% of questions in every bucket in both domains. Beyond that it falls to between 50% and 62%.

The mechanism is arithmetic. A count over 442 rows is wrong if one row is wrong. On the people corpus the failures at high breadth are mostly counts that came out one off: 495 against a true 496, 441 against 442, 49 distinct countries against a true 50. On the organisations corpus the pattern is sharper. The table missed 11 of 56 questions; ten of them ask about founding year and all ten trace to the same eight documents where Deasy declined to state one. The eleventh is the 2015 tie that also defeats the ceiling. Not one of the 11 asks about headquarters location, the field with 100% agreement and no gaps.

Document reading degrades over the same range too, and not uniformly. On organisations it falls from 83% on questions with no matching documents to 25% at 21 to 60, faster than the table does. On people it does not: at 21 to 60 documents reading answered 83% against the table's 50%, on 6 questions. That is one of the few places where reading is clearly ahead, and on a sample that small it is a hint rather than a result.

Coverage on one corpus, correctness on the other



Where Deasy committed to a value it agreed with the independently verified answer between 98.6% and 100% of the time. This chart measures correctness only: documents where Deasy declined to commit are excluded from the denominator and reported separately as coverage, because they are a different failure. Note the truncated axis, which starts at 95%.

Where Deasy committed to a value, it agreed with the verified answer as follows: headquarters location 100.0%, founding year 99.6%, birth country 98.6%, birth year 100.0%, country of death 99.7%. This is the measurement that describes the metadata, taken directly against answers derived independently of it. Read the denominator carefully, though. These are rates over the documents where a value was produced, and the documents where Deasy declined are excluded. Declining and being wrong are different failures with different fixes, which is why they are separated, but it means the chart shows only one of the two.

On the organisations corpus the other one is the whole story. The table has exactly eight rows where Deasy gave no founding year, and those eight account for ten of the eleven remaining failures. You can see them in the answers: a median of 1967 against a true 1968, a count of 418 companies founded before 2000 against a true 424, an average founding year for California of 1979.4 against a true 1978.7. In each case the query was exactly right and the data underneath it was incomplete. On

this corpus, **coverage rather than correctness is the binding constraint**, and the ceiling shows what closing those eight gaps would be worth: 45 of 56 becomes 55 of 56.

The people corpus does not follow that pattern, and saying so matters more than the tidy version would. There Deasy declined only once, yet the table still missed 14 of 56, with counts off by one to six. That is what roughly seven wrong birth countries out of 500 produce when you aggregate over them. On that corpus the binding constraint is correctness. One corpus each way is not enough to say which dominates in general.

Threats to validity

Scope is the largest limitation. Two domains, 56 questions each, one model, one agent framework, one tagging system (Deasy), one table of two or three columns and no joins. Treat the mechanism as the transferable part and the numbers as specific to this footprint. The flat-cost line in particular depends on the schema being small enough to hand an agent whole; at enterprise scale, finding the right columns becomes its own retrieval problem that this study does not test.

The questions were generated from the columns the table holds. This is the largest reason the 45 of 56 will not transfer as a number: it removes an entire failure mode that every real deployment has, it biases in our favour, and it is not small.

The corpus was pre-filtered to documents a rule-based parser could resolve unambiguously. Entities whose attribute was ambiguous or split across two values were dropped. That keeps the scoring fair between configurations, which is why it was done, but it removes exactly the documents where extraction is hard and where a reading agent's judgement might beat a scalar column.

The table's build cost is not measured. No tokens, no wall clock, no human hours for choosing fields and validating output, no refresh cost. Every cost figure here is marginal cost per question at query time.

The corpora are public encyclopedia content, well represented in pre-training. The floor measures the effect directly and question scoping limits it, which reduces the concern considerably without removing it. A study on private corpora would be strictly better evidence.

The reference answers were never validated by a human. They come from a deterministic parser, and the audit that checks them recomputes rather than independently re-derives. Wherever that parser is wrong, document reading and Deasy are both penalised while the ceiling configuration is immune, because it is built from the parser's own output.

A null is not proof of absence. On the people corpus both configurations answered 42 of 56. The honest statement is that there is no accuracy difference large enough for this design to detect, not that there is no difference. An agent's answer to a broad question depends on how exhaustively it happens to search, so a 56-question set cannot resolve a difference of a few questions in either direction, and no such difference should be read as a result.

This measures the value of a metadata table, not the day-one output of a product. Every conclusion is scoped to metadata of the quality stated field by field above. A reader whose own extraction is

materially worse should expect materially worse results, and the ceiling shows what better extraction would be worth.

The questions are machine-generated and phrased more precisely than a person would phrase them. Real users ask messier, vaguer questions, which adds an intent-resolution problem this study does not test.

Scoring is strict. One missing name in a list of sixty fails the question, and a count is wrong if one row of hundreds is wrong. On the organisations corpus, where the table's residual defect is missing values rather than wrong ones, that penalty falls on the configuration under test and probably understates its practical usefulness. It is not a one-way bias in general: strict scoring also denies document reading credit for answers that were nearly right.

Cost figures are token-based. Token counts include cache reads at par with fresh input, which is right for measuring work done and wrong for predicting a bill. Dollar amounts are list-rate estimates for one model at one point in time.

A third approach was not tested. Between a precomputed table and an agent grepping ad hoc sits an obvious middle: extract the field from each document on demand, then aggregate. It needs no tagging step and would likely be accurate at bounded cost. We did not run it, so what is demonstrated is that precomputed structure beats unstructured searching, not that it beats structured extraction on demand.

What this means for you

The shape of the workload decides whether this pays, and the mechanism tells you the shape.

Metadata pays when questions are broad and the corpus is large. Anything phrased as how many, which is the most, what is the average, or list all of them, over more than a few dozen documents. The table's cost is flat in the number of documents; reading's is not. That gap was 3 to 8 times at the narrow end of what we measured and 14 to 48 times at 21 to 60 documents. Do not extrapolate it as a straight line: our own widest bucket came back down.

Work out your own break-even. This study measures query-time cost only. The arithmetic you need is: excess tokens per reading question, times the aggregate questions you expect, against the cost of tagging the corpus once and keeping it current. Our excess was about 107,000 tokens per question on one corpus and 357,000 on the other, so payback arrives fast at any real query volume, and never on a corpus tagged monthly and queried twice.

Metadata does not pay for narrow lookups. Find the clause, summarise this report, what does this document say about X. Reading is cheap and accurate when the answer sits in one place. This is not a replacement for retrieval; it is the other half of a workload retrieval was never designed for.

Pick the fields before you tag. A table can only answer questions about the columns it has, and on the organisations corpus every failure but one came from a single column with eight holes in it. The fields worth extracting are the ones your aggregate questions filter and group by: places, dates, categories, status values.

Measure how often the extractor declines, not just how often it is wrong. Per-field accuracy where a value was given was 98.6% to 100% in both corpora, yet one corpus was limited by the eight documents with no value at all and the other by the handful with a wrong one. For aggregate queries the two failure modes are equally fatal and they call for different fixes, and an accuracy figure computed over committed values shows you only one of them.

Ask where the answer came from. A count from a table arrives as a number with no trail behind it, and a stale or wrong cell fails silently and confidently. The reading configuration at least produced the list of companies it counted, which a person can spot-check. If answers must be defensible, budget for provenance. The same goes for permissions: a flattened table aggregates across documents whose access rules may differ, and a count does not respect them by itself.

Methods appendix

Agent and environment. Claude Sonnet, model identifier `claude-sonnet-4-6`, driven by Claude Code as an autonomous tool-using loop. Identical model, prompt structure and per-question spending ceiling of $2.00 across all four configurations, four questions in flight at a time, and no question terminated by the ceiling; the most expensive single question anywhere cost $1.79. The document-reading configuration ran in a sandbox containing only the corpus, with shell, file-read, text-search and glob tools and no network access. The table-backed configurations reached a PostgreSQL database holding one table, through a query tool and a schema tool, with no access to the documents. The no-information configuration had no tools.

Corpus construction. Entities were seeded from Wikidata, restricted to those with an English Wikipedia article. Each article was kept as one markdown file combining the article's summary fact box with the plain prose, since the target values very often appear only in the fact box. A deterministic parser then read the fact box first and the opening prose second, with rules for the recurring traps: exact token matching on comma-separated place names, so that "North Kansas City, Missouri" does not resolve to Kansas; stripping county names, so that "Washington County, Oregon" does not resolve to Washington; and cutting at clause boundaries, so that a sentence mentioning a subsidiary's separate headquarters yields nothing rather than the wrong answer. Entities the parser could not resolve unambiguously were dropped from the corpus. For the people corpus, a curated settlement-to-modern-country map resolves birthplaces in multi-successor polities, so that a birth in Prague under Austria-Hungary resolves to the Czech Republic rather than to Austria. The people corpus was then sampled deterministically to 500 with a committed random seed, so the same pipeline reproduces the same corpus.

Question generation. Every question and every reference answer is computed from the verified attribute data, never hand-written, and the set regenerates. The 11 operator families are: count with one filter; count with two filters; count whose true answer is zero; count of distinct values; earliest and latest across the corpus; earliest and latest within a named group; average across the corpus; average within a group; median; the group with the most members; and list every member of a group. Questions whose true answer is none make up 10.7% of the organisations set and 16.1% of the people set. Breadth spans zero documents to the whole corpus.

Scoring. Beyond the rules given above, three parsing conventions keep the scorer from blaming the agent for the scorer's own mistakes: read the answer from the last standalone number rather than the first number in the text; discard list items longer than sixty characters or seven words, since those are the agent explaining itself rather than naming an entity; and match entity names by containment after normalising, so that "Truist" and "Truist Financial" both count. Absolute and relative error is recorded alongside every pass or fail, because a near miss and a guess are different failures.

Validation. An audit recomputes every reference answer and rechecks the table, the questions and the scorer; both corpora pass with zero errors and zero warnings. This is an internal consistency check, not an independent re-derivation, and no human sampling was done. The gold-table ceiling is the end-to-end version of the same check: perfect SQL over the table answers 98.2% and 100% of the questions.

Metadata source. The corpus was ingested and tagged by Deasy's document-tagging product against a schema of two fields for the organisations corpus and three for the people corpus. The resulting tags were joined by document identifier into the single table the agent queries. No value in that table comes from any other source, and the table-backed configurations had no access to the documents to fall back on.

Metadata quality. Agreement between each value Deasy extracted and the independently verified answer, over the documents where Deasy committed to a value: headquarters location 100.0%, founding year 99.6%, birth country 98.6%, birth year 100.0%, country of death 99.7%. Documents where Deasy declined are excluded and reported as coverage instead. Country of death carries an extra caveat: a null means "still living" in this schema, so an abstention on a laureate who has died is indistinguishable from a correct null and drops out of the denominator.

Cost accounting. Tokens per question counts all input and output tokens across every turn, including cache reads, so it measures work done rather than what a billing arrangement would charge. Dollar figures are list-rate estimates for one model, reported for order of magnitude only.

What ships with this report. The plotting script, `make_figures.py`, and its style module `deasy_mpl.py`, so every figure can be regenerated from the numbers stated here. The corpora, question sets, reference answers and per-question traces are not included, which means the figures are reproducible from the results but the results are not independently reproducible from this document alone.