Optimising for Memory Recall

A better model was available at the same size and we didn't ship it. Ten encoders, five corpora, six compiler settings and fifty quantisation configurations later — here is what the megabytes actually bought, and why the headline benchmark was the wrong scoreboard.

Kaleidoscope is memory for AI agents, and it runs entirely on your laptop. No API call, no inference server, no network. That constraint is the whole design, and it turns every quality decision into a budget decision: the model ships *inside the binary*, so every point of retrieval quality is paid for in megabytes a user downloads.

This is what that budget actually bought.

Starting from something free and bad

The first version used no model at all — a hashed projection of tokens into 192 dimensions. Zero bytes, no download, no inference.

It was also last. On BEIR SciFact it scored **0.310** nDCG@10 and on NFCorpus **0.130**, behind every real model we tested it against.

There was a tell, visible without any benchmark: **84 of its 192 dimensions were zero.** Forty-four percent of the vector was empty, every time. We were paying to compare bytes that carried nothing.

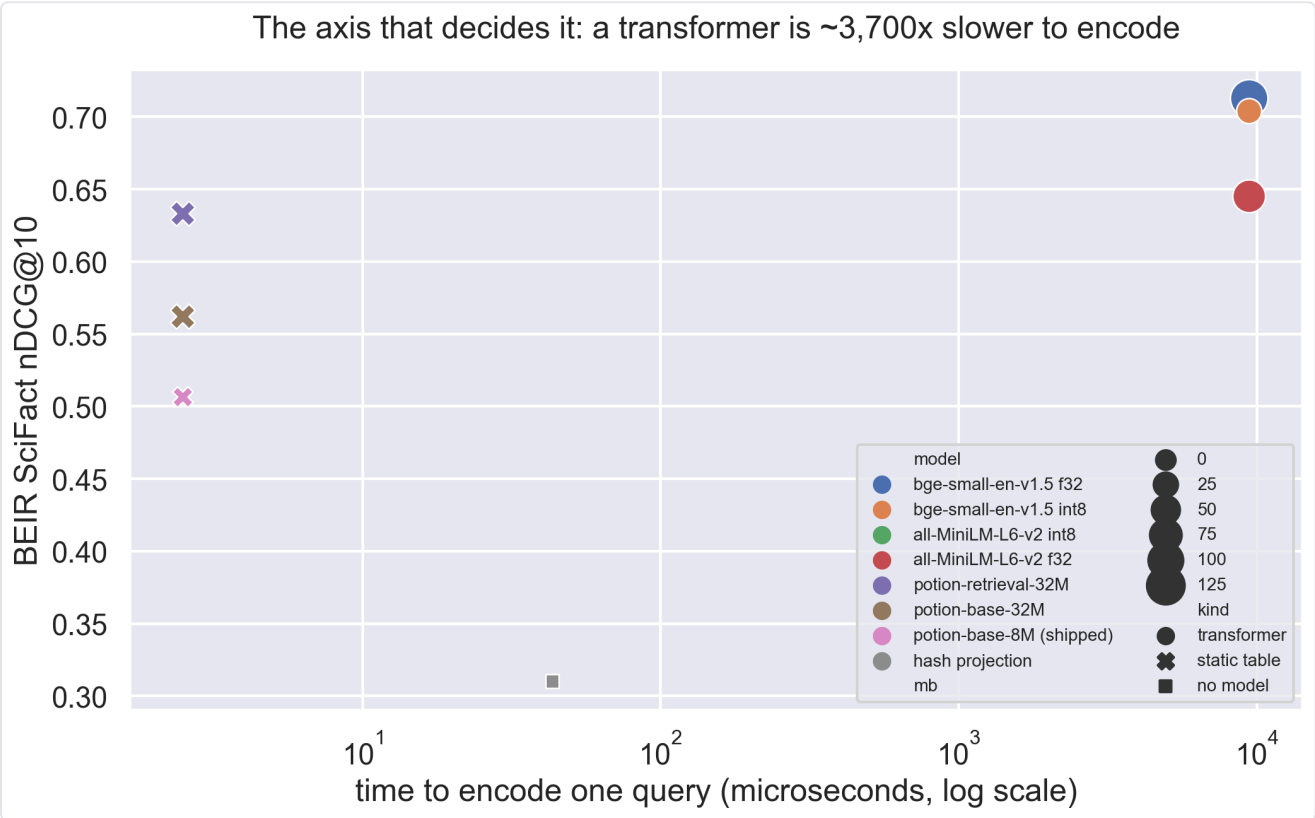
Ten candidates, and an uncomfortable chart

We measured ten encoder configurations on two public retrieval corpora nobody here authored.

what each model costs to ship against what it retrieves

Read on size and quality alone, **our choice looks wrong**. `bge-small-en-v1.5` quantised to int8 is **31.8 MB** and scores **0.704** on SciFact. `potion-retrieval-32M` is **30.8 MB** and scores **0.633**. Same shelf space, seven points better. On those two axes potion is dominated outright.

We ship the potion family anyway, and the reason is a third axis the size chart can't show.



retrieval quality against the time to encode one query

`potion` is a lookup table: you fetch a vector per token and average them. One query takes **2.5 µs**. `bge-small` is a transformer — it runs an actual forward pass, and one query takes roughly **9 ms**. That is about **3,700×**.

For a memory system that is not a detail. Every recall encodes the query, and encoding is on the path before any comparison happens. Nine milliseconds per query is survivable; nine milliseconds inside a loop is not.

There's a second, harder blocker. The transformer needs a real tokenizer, the Rust `tokenizers` crate refuses to compile without `fancy-regex` or `onig`, and both are on our deny list — first-party code decides what a value is by parsing it into a type, never by matching its text. Shipping bge meant either abandoning that rule or writing a tokenizer. (We did eventually write one, which is its own story.)

So the choice is not "best nDCG per megabyte." It's "best nDCG per megabyte, among encoders that answer in microseconds and don't drag in a regex engine." Within that set, size is what's left to trade.

Then, inside the family, halving it again

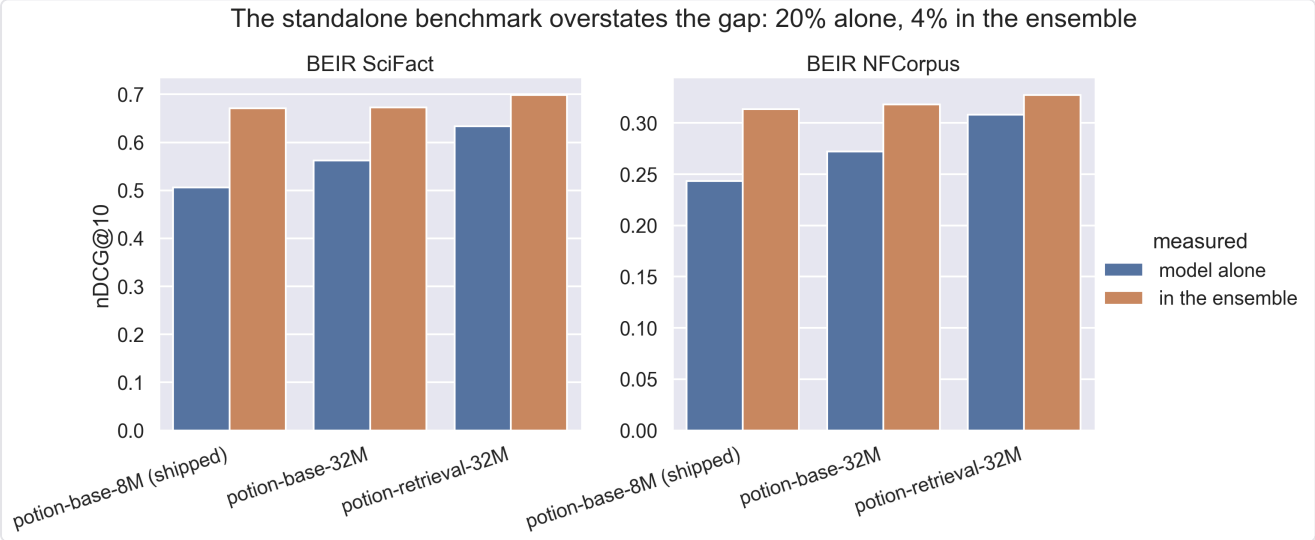
`potion-retrieval-32M` took the release binary from 25 MiB to **48 MiB**. `potion-base-8M` costs **7.2 MB** — a quarter of the 32M models — and gets **0.506** against 0.633.

So: give up 20% of retrieval quality, halve the binary. We took it, and the release cap went back to 24 MiB.

Twenty percent is a real loss and worth saying plainly. It is also the wrong number, and it took five corpora to find that out.

The standalone benchmark is not the scoreboard

The model never answers a query by itself. It is one input to an ensemble, and what matters is the ensemble's output — so we measured that, choosing the ensemble's configuration on a **training** split and reporting it on a **held-out test** split, so nothing here is tuned on the number it reports.



each model alone, and the same model inside the ensemble

BEIR SCIFACT	ALONE	IN THE ENSEMBLE
potion-base-8M (shipped)	0.506	**0.671**
potion-base-32M	0.562	0.673
potion-retrieval-32M	0.633	0.698

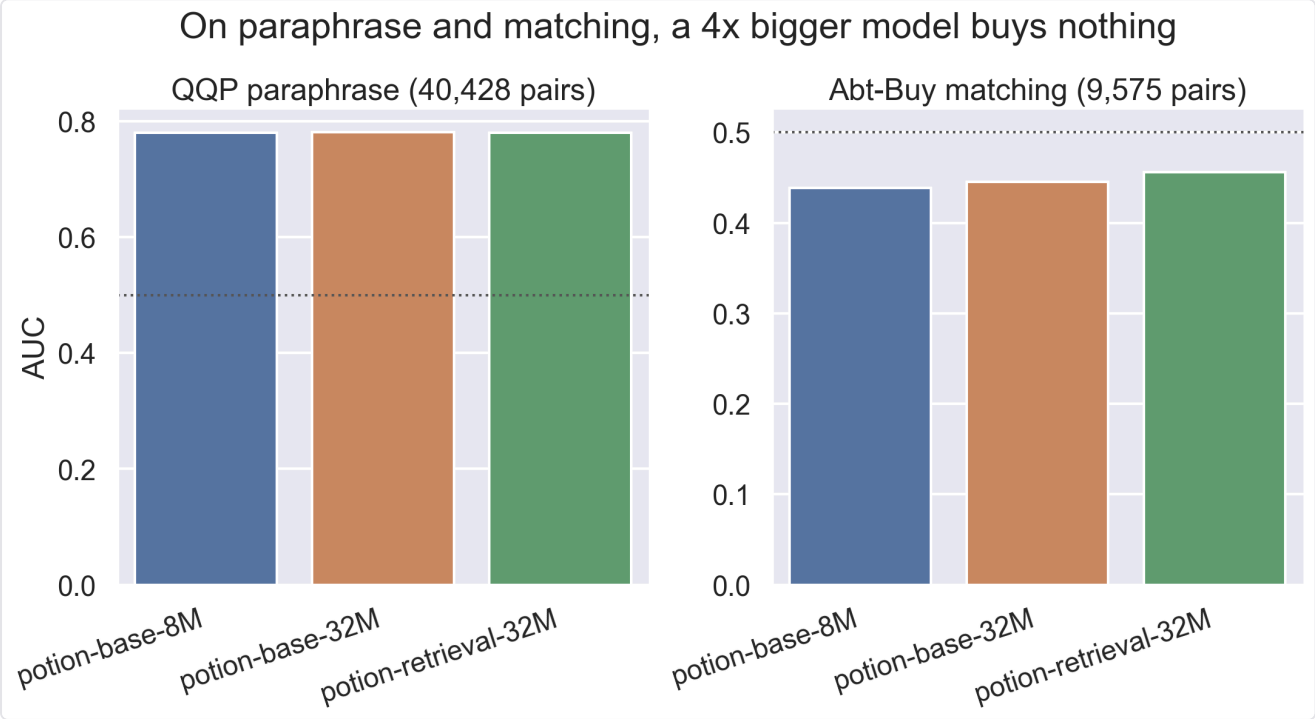
Alone, the 8M model retains **80%** of what the 32M retrieval model delivers. Inside the ensemble it retains **96%**. NFCorpus agrees almost exactly — 79% alone, 96% ensembled.

So the 20% deficit is real in isolation and about **4%** where it is actually felt. The benchmark overstated the cost of the cheap model roughly five-fold.

There is an uncomfortable corollary we should own. When we swapped models, we did **not** re-fit the ensemble's configuration — it had been fitted for the previous model five days earlier. Measured on external data, the un-refitted ensemble was *worse than not using the model at all*. The model was fine; the thing around it was calibrated for a predecessor. Refitting is what produced the numbers above, and "re-fit the ensemble when you change the model" is now a step rather than a hope.

Where the gap disappears completely

Retrieval is one job. We also ran two corpora nobody here authored, on the matching task deduplication actually performs.



paraphrase identification and product matching

On **QQP**, 40,428 labelled paraphrase pairs, the three models score **0.7801**, **0.7810** and **0.7799**. That is a spread of 0.001 across a four-fold difference in model size — below the standard error.

The cheap model is not slightly worse here; it is indistinguishable.

On **Abt-Buy** product matching, all three score **below 0.5** — worse than chance. When an entire modality fails, its size is not the interesting variable.

So of five corpora, the 8M model is behind on two, at par on one, and irrelevantly bad alongside everything else on another. The one number that would have made us buy 24 more megabytes was the one measured furthest from the job.

Three optimisations that were free

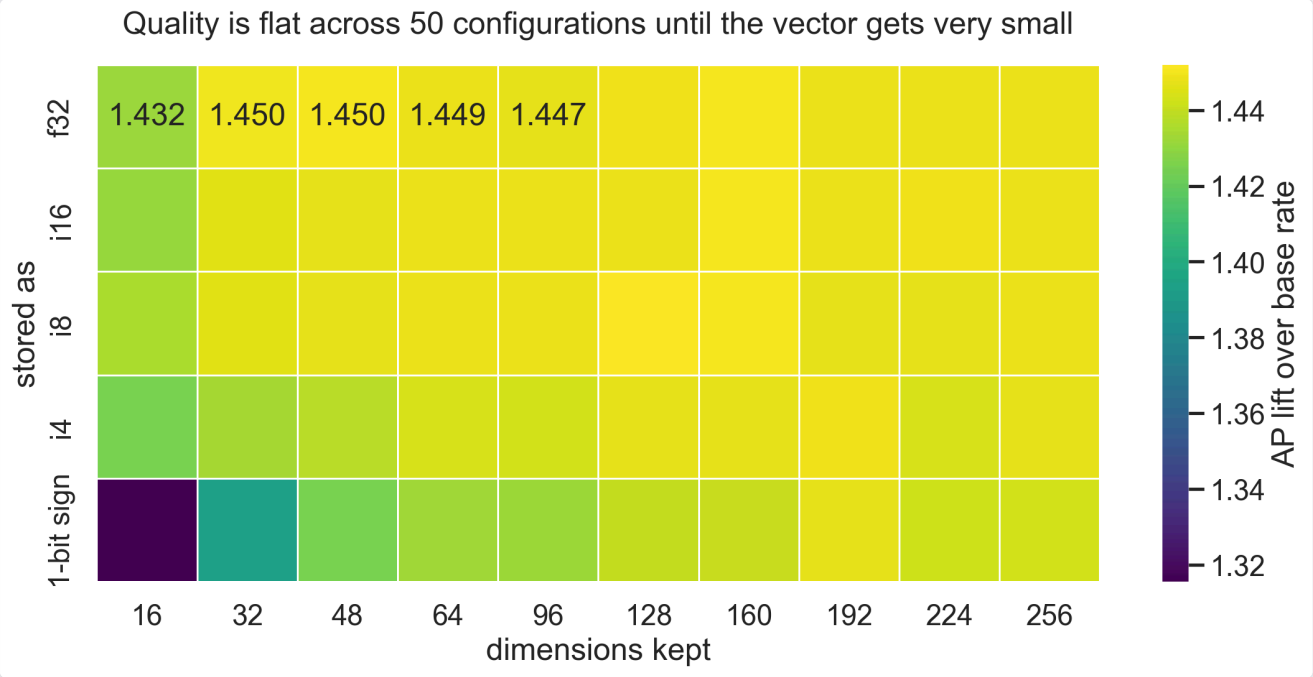
With the model chosen, the remaining question was how much could be recovered without giving anything back. Three things cost nothing at all.

Hold the table at I8. The weights ship quantised to 8-bit. You can widen them to f32 once at load and compare in float, or leave them narrow and compare in place. In place: **7.2 MiB** resident. Widened: **28.8 MiB**. Encoding speed is the same either way — 13.8 µs against 13.4 µs.

And the output is not merely close. Across the test texts, **zero bits differ**. A 4× memory saving with provably identical vectors is not a trade-off; it's just the better option.

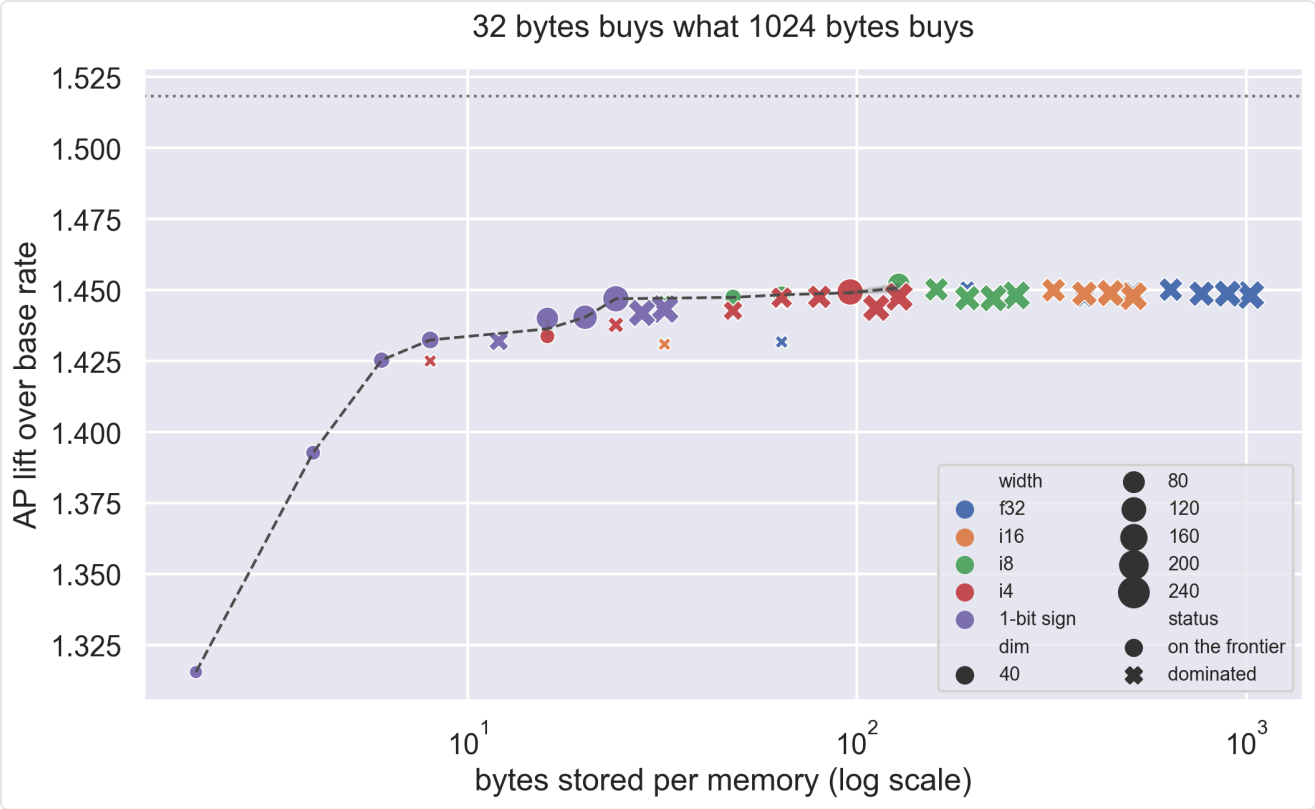
Store i16, not f32. potion emits f32; the vault stores i16 and the read path compares i16. f32 costs **180.56 ns** per comparison against **30.40 ns** for the same vectors — **5.94×** — and twice the disk.

The obvious worry is that rounding costs ranking quality. So rather than check the one configuration we ship, we swept **fifty**: five widths from f32 down to a single sign bit, each at ten dimensions, scored on 1,040 hand-labelled pairs.



average-precision lift across all fifty configurations

The surface is almost flat. Everything from f32/256d down to **1-bit/96d** sits between 1.432× and 1.452× against a ceiling of 1.518×.



the same fifty points, plotted against what they cost to store

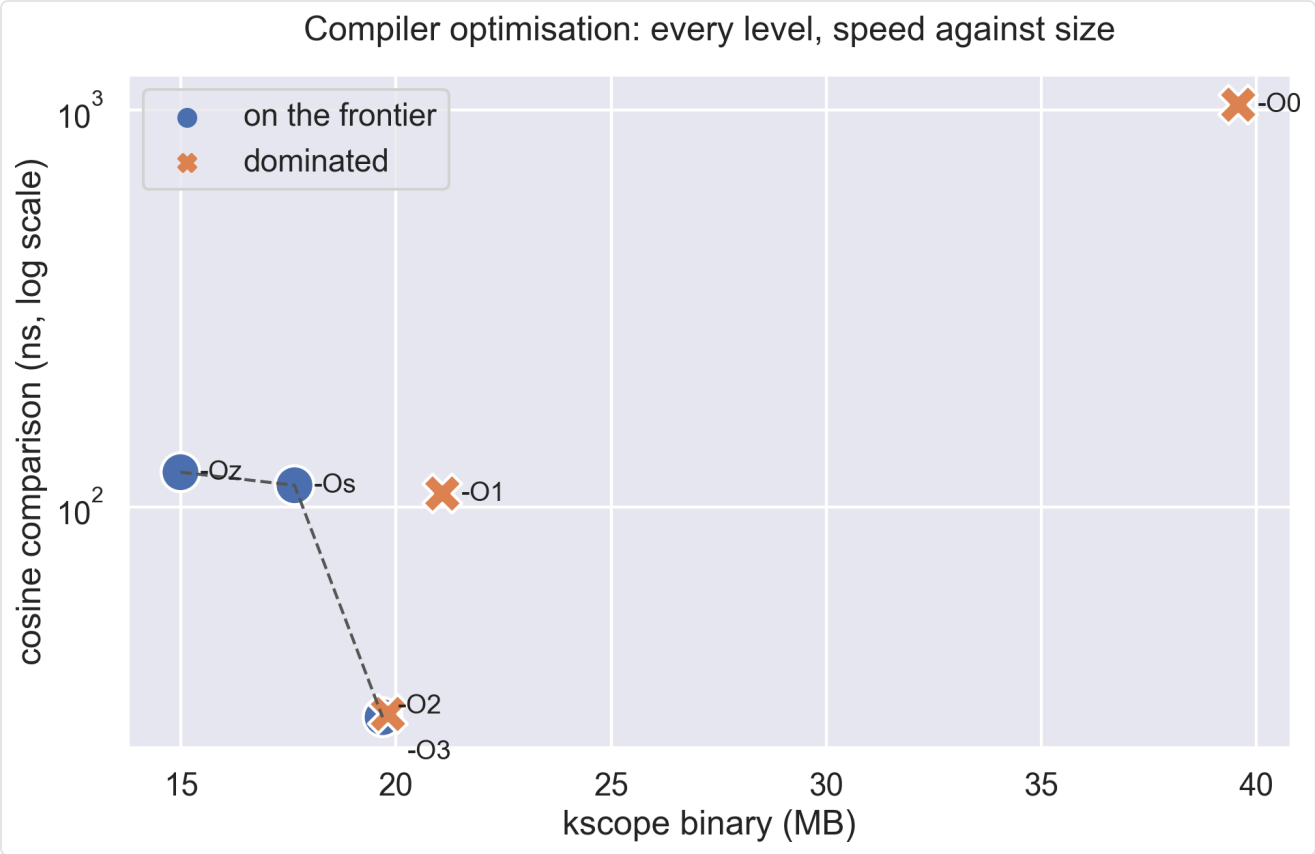
That's the frontier, and the knee is much further left than we expected. **32 bytes per memory scores 1.443×. 1024 bytes scores 1.448×.** A 32× reduction costs about a third of one percent.

Quality only breaks when the vector gets genuinely tiny — 1-bit at 32 dimensions (4 bytes) drops to 1.393×, and at 16 dimensions to 1.315×.

Two honest caveats. potion is **not** a Matryoshka model, so a truncated prefix isn't trained to stand alone; that it survives truncation this well is a property of this corpus, not a general licence. And this measures a *duplicate-detection* decision, not a retrieval ranking — the two ask different things of a vector.

We ship i16/256d. On this evidence that is conservative by a wide margin, and the sweep is the argument for revisiting it.

Don't use `-O3`. This one genuinely surprised us.



every compiler optimisation level, speed against binary size

LEVEL	BINARY	COSINE
`-O0`	39.6 MB	1029.94 ns
`-O1`	21.1 MB	107.99 ns
`-O2`	**19.7 MB**	**29.51 ns**
`-O3`	19.8 MB	30.06 ns
`-Os`	17.6 MB	113.32 ns
`-Oz`	15.0 MB	122.24 ns

`-O3` is **bigger and slower than** `-O2` — dominated on both axes, so there is no configuration in which it is the right answer. `-O0` and `-O1` are dominated too. The frontier is only three points wide.

The 35× gap between `-O0` and `-O2` is also worth noticing for a different reason: it is larger than any algorithmic change discussed in this post. A kernel timing without its optimisation level attached is not a measurement.

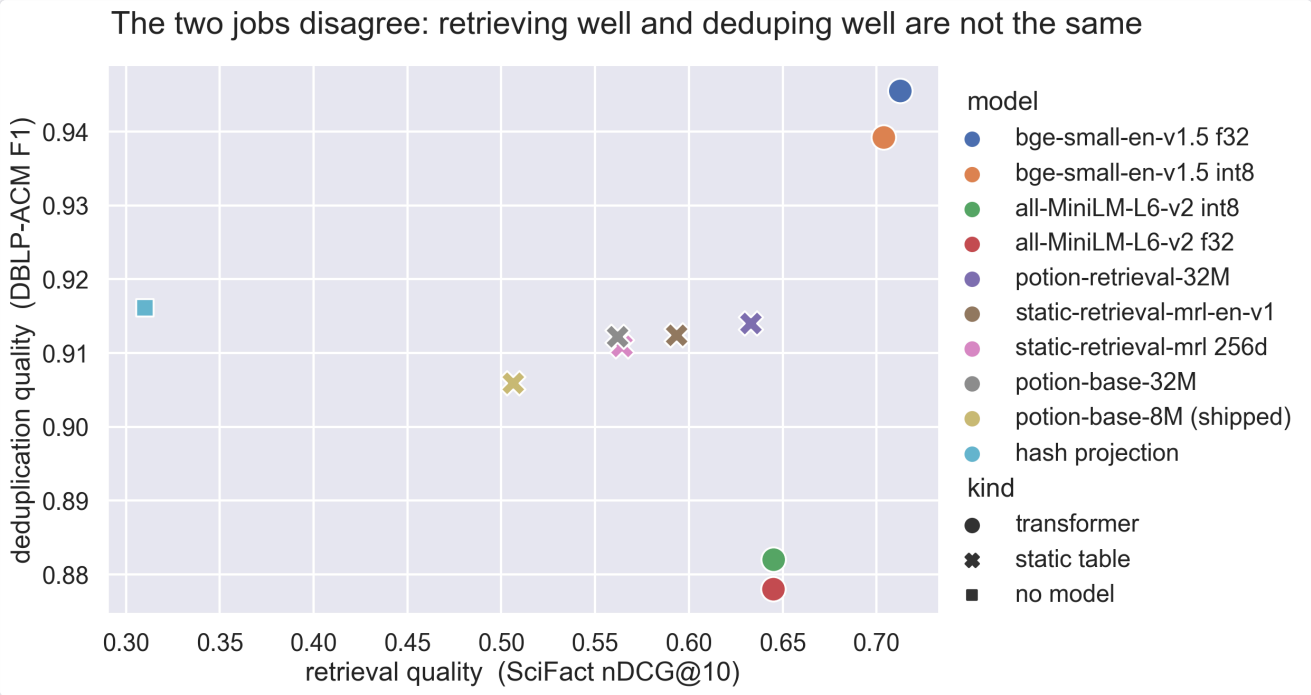
The one that wasn't free

`-Oz` is a real decision. It saves **4.7 MB** of binary — 24% — and costs **4.1×** on the comparison kernel.

We ship `-O2`. For a memory system the scan runs on every write and every read, while the download happens once. But if you're shipping to a phone, or the binary is the thing users complain about, `-Oz` is defensible and sits on the frontier. That's what a frontier is for: it tells you which options are real, not which one to pick.

The result that reframed the problem

While comparing models we ran them on entity resolution as well as retrieval — matching records that describe the same thing, which is what deduplication does on every write.



retrieval quality against deduplication quality

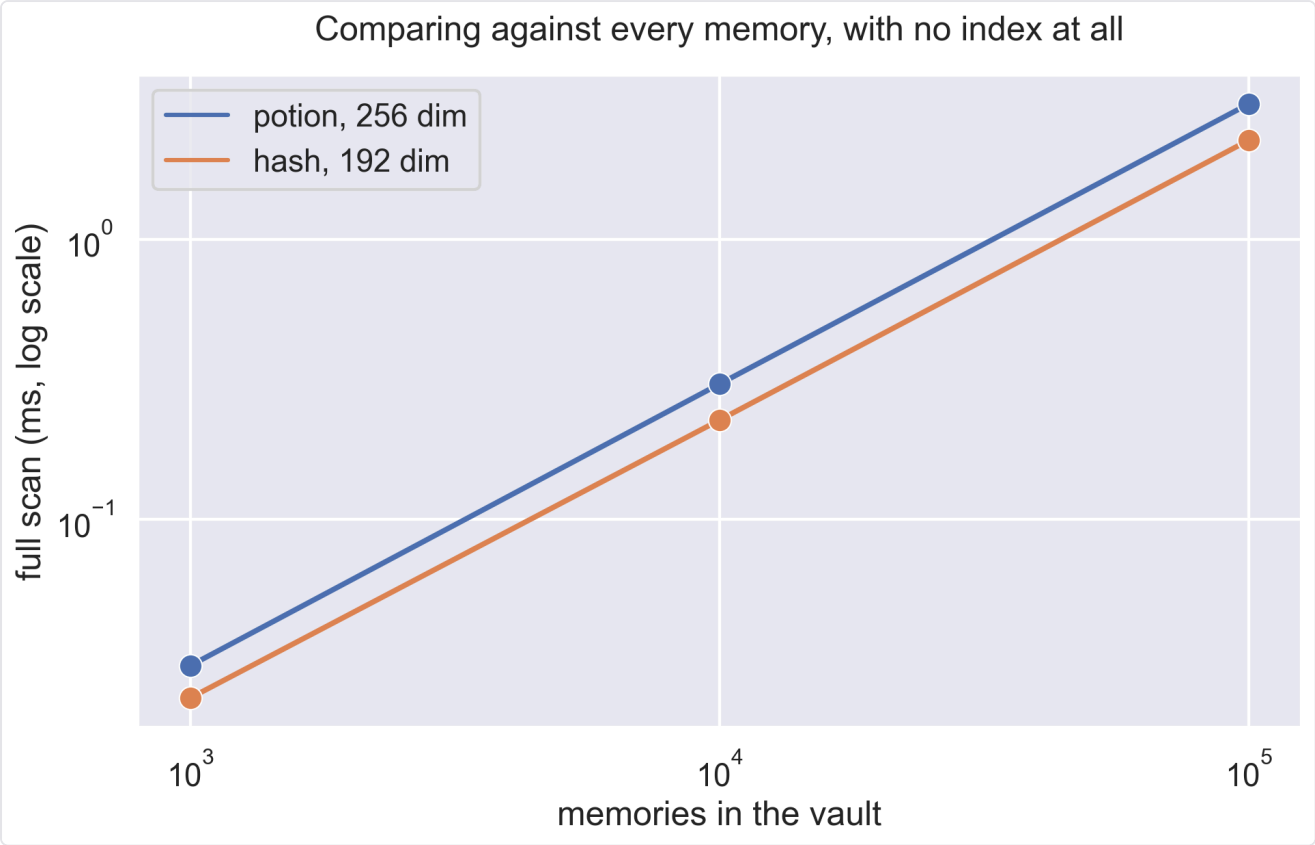
The ranking inverts. On DBLP-ACM the **hashed projection** — the free one, last on both retrieval benchmarks — reaches **F1 0.9161**, beating every potion model and both `all-MiniLM-L6-v2` configurations by 3.8 points. Only bge beats it.

The model that retrieves worst dedupes best.

In hindsight it isn't mysterious. Retrieval rewards understanding that "cycle credentials monthly" and "rotate keys every 30 days" are about the same thing. Deduplication requires noticing that "every 30 days" and "every 90 days" are *not* the same thing, and a model trained to collapse surface differences will collapse exactly the difference that matters. Semantic similarity is a recall instrument, not an identity test.

So the two jobs get different signals. That's not a compromise; it's the correct architecture, and we only found it because we benchmarked one thing on a task it wasn't chosen for.

What it adds up to



full scan cost against vault size

Comparing a new memory against **every** memory in a 100,000-entry vault costs **3.04 ms**. No index, no ANN, no tree, nothing to keep in sync and nothing to rebuild.

That's the part worth taking away. We spent a long stretch earlier looking for a cleverer data structure to avoid comparing against everything. The representation work made comparing against everything cheap enough that the structure was never needed — 30 ns per comparison turns a scan into a rounding error.

Make the unit of work cheap before you build machinery to avoid doing it.

What we're not claiming

- **None of these corpora are agent memories.** SciFact, NFCorpus, QQP and Abt-Buy are public, which is exactly why we used them — nobody here authored them, so nobody here could tune to them. They are still not our workload, and the ensemble result should be re-measured on real memory traffic before it is treated as settled.
- **The pair corpora are scored with AUC**, which is what that harness emits and not the metric we prefer; it is prevalence-invariant, and a false merge is far more costly than a missed duplicate. Both conclusions above survive anyway — one is a spread below the standard error, the other is a direction rather than a threshold — but the numbers deserve average precision before they carry more weight than that.

- **These are single-machine numbers**, on an M4 Pro. The ratios should travel; the absolute nanoseconds will not.
- **There is no `potion-retrieval-8M`.** The retrieval-tuned variant is published only at 32M, so the cheap-and-retrieval-tuned corner of this grid does not exist to be tested.
- **The quantisation sweep is one corpus and one task.** 1,040 labelled pairs on a duplicate-detection decision, not a retrieval ranking. It says rounding is nearly free for *this* judgement, and potion is not a Matryoshka model, so the truncation result is not a general property.
- **Transformer encode latency is a re-used measurement**, taken on this same machine on 2026-08-08 under torch rather than ONNX. It is an order-of-magnitude figure — the 3,700x gap is not sensitive to the third digit, but do not quote it as one.
- **AP is reported on the natural slice only.** This corpus is deliberately half adversarial, so a pooled figure would measure how much adversarial data was authored rather than how good a signal is.

Kaleidoscope is the agent memory layer we're building at Kleos. Every figure is generated from a harness that runs against the production code path — if the system changes, the numbers move with it.