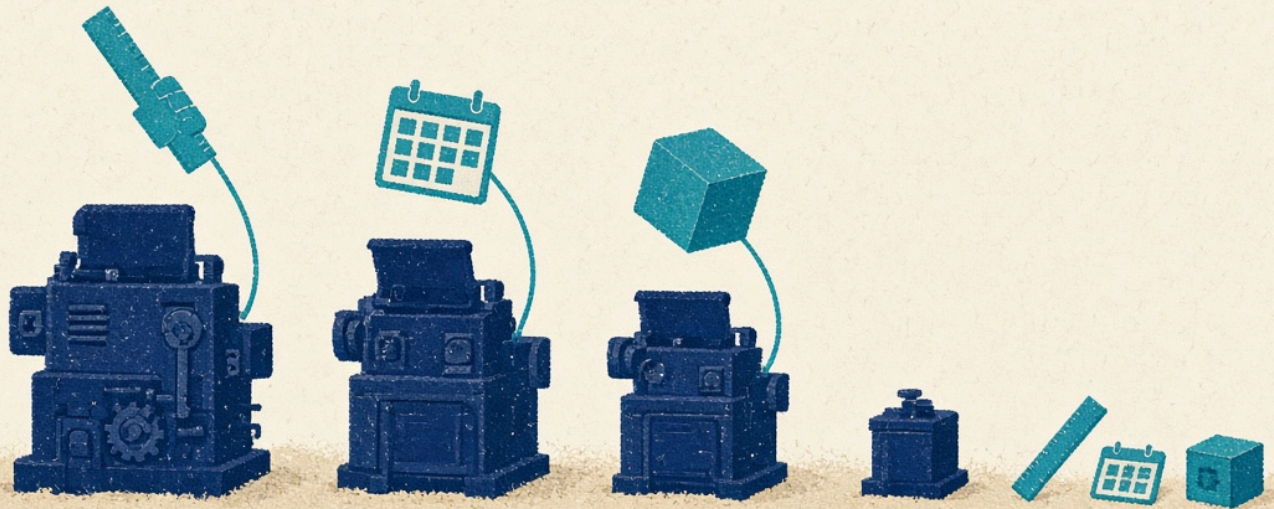




SMALL MODELS · CRAFT

Take jobs away from the model

A tiny lab keeps finding the same thing: models get smaller and more reliable every time you give one of their jobs to plain code.

BY MAGAZINER · OCTOBER 2026

IN SHORT

Haku Lab builds models measured in megabytes that run on a laptop in milliseconds. Its write-ups share one design rule (let the model do the fuzzy part and let ordinary code do the exact part) and one warning: the metric can lie. A prototype that reads a transcript straight out of an embedding model shows how far the idea can go.

MOST TALK ABOUT AI is about making models bigger. Haku Lab, a tiny independent lab, keeps publishing evidence for the opposite move. Its models are a few megabytes, do one narrow job and run on a Mac in milliseconds with no network.

The thread through all its work, I think, is one idea: a model gets smaller and more reliable every time you take a job away from it. And the lab's other habit, publishing what broke as well as what worked, is the reason to trust the idea.

All the figures in this essay come from the lab and its developer themselves. None has been checked independently.



1

LET CODE DO THE EXACT PART

The cleanest example is dates. Language models are surprisingly bad at phrases like “next Tuesday at six” or “end of the quarter.” So Haku’s 14-million-parameter model doesn’t calculate anything. It only turns the phrase into a small structured form, like “weekday, next, Friday, 15:00.” A few lines of ordinary code do the calendar arithmetic. The model never counts days, so it can be tiny and still reliable. (The lab notes a few constructions still trip it.)

A web element classifier works the same way. One Amazon page had over 13,000 page elements and about 1,800 you could interact with. A 9-million-parameter model sorts each into one of 25 intent types, so the checkout button wins even when its label reads like a newsletter link. On fourteen sites it never saw, it was right about 85 percent of the time, in a 2.5-megabyte bundle. The list of types came from clustering the data first, not from a hand-written list.

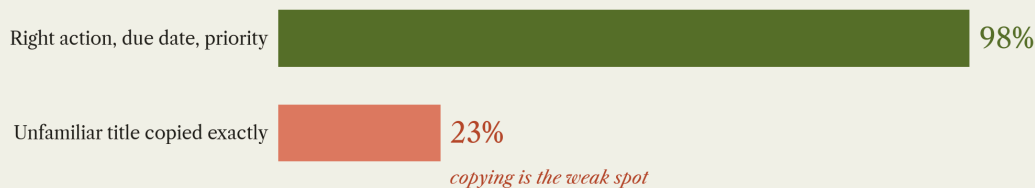


2

KNOW WHAT TINY MODELS CAN'T DO

The lab is just as clear about limits. Its 25-million-parameter model for turning requests into structured tool calls picked the right action, due date and priority about 98 percent of the time on unseen requests. But copying an unfamiliar title back word for word worked only about 23 percent of the time.

Tiny models classify well and copy badly



Source: Haku Lab, 25M-parameter tool call model on unseen requests. Not independently verified.

That gap is the whole lesson in one chart. Classification fits in a tiny model. Open-ended copying doesn't. So if a job needs exact text, don't ask the model to retype it. Copy it with code.

*A model gets smaller and more reliable every time
you take a job away from it.*



3

START FROM SOMETHING PRETRAINED

A second pattern: keep a frozen pretrained model and train a small piece on top. Haku's end-of-turn detector (which tells a voice agent when you've stopped talking rather than paused) plateaued at about 30 percent recall when trained from scratch, across three tries. Part of the problem was a label bug. The bigger fix was switching to a small adapter on an existing 230-million-parameter model. That lifted turns ended at the right moment from about 41 to 60 percent, at a cost of 10 to 30 milliseconds per check instead of one.

The same move powers the most surprising prototype. A developer, TSL Shahir, took Google's EmbeddingGemma 2, which places speech and text on one shared map of meaning, and left it untouched. On top he trained a small reader that moves each slice of audio onto the text side and looks up the nearest of about 229,000 English words. That's a dictionary lookup, not a large model writing text.

He reports an hour of audio transcribed in about 13 seconds on a base MacBook Air, with a word error rate around 5 percent, against 2 to 3 percent for leading speech models. It's English only for now, users reported poor results in other languages and on songs, and there's no paper yet. ("Skipping the transcript" explains the shared map it relies on.)



4

THE METRIC CAN LIE

The habit I'd most like to copy is Haku's suspicion of its own numbers. Three examples.

A faster text-to-speech method sounded buzzy, but the word error metric didn't notice. The listening test was right.

The lab tested a large compressed model on 21 traps for agent failures. It passed most, but failed reconciliation every time: it read a March wiki page saying 32 percent margin and never read the June note revising it to 28, sitting in the same search results. Function-calling benchmarks don't catch that.

And a model that reasons before answering sometimes left its answer in the reasoning. Score only the final field, the lab notes, and you're measuring your own truncation.



5

WHY THIS MATTERS BEYOND ONE LAB

This connects to other essays. My earlier essay ["Who gets to move the knowledge"](#) shows Coinbase winning with a small model trained on its own labelled data. "The factory is the record" argues the checks matter more than the code. Haku adds the craft: split the job, let code handle the exact parts, and distrust a single number.

The strategic point is cost and privacy. If a 2.5-megabyte model can route web elements and a laptop can transcribe an hour in seconds, some of what companies now pay for per token could run on the user's own machine, with their data never leaving it. I don't know how far that goes. But every time Haku takes a job away from a model, the model gets smaller, and it doesn't seem to get worse. ■

Sources: Haku Lab's write-ups (lab.haku.sh); TSL Shahir's posts on EmbeddingGemma Transcribe (12 October 2026). None of the numbers are independently verified.

Illustrations and charts were made for this article. Set in Libre Caslon and Josefin Sans.