Cactus Needle

An open 14MB model for tool calling, device use, and structured extraction.

NEEDLE 2 SANDBOX

loading model...

DEFINED TOOLS

QUERY

ask for something the tools c

Run

RESULT

TRY AN EXAMPLE

Smart home · multi-step

Robot · 3 moves

Gallery · chained

Device control · 3 actions

Extract → email

Currency · live API

Document · extraction

Sentiment · classification

12 tools · routing

Repeated calls

Array argument

Flight · form filling

Off-topic · refusal

Running in WebAssembly directly in your browser

Today we release Needle 2: an open 45M-parameter model for tool calling, device use and structured extraction. The whole model is a single 14MB binary that runs a full session in 28MB of RAM. It is built on our Simple Attention Network findings, compressed to CQ2-bit with Cactus Quants, and baked into its own engine.

On the tool call and mobile device use benchmarks, Needle 2 trades wins with other small models like FunctionGemma 270M, LFM2.5 230M and Apple FM, at 5× to 70× smaller, and 2 bits against their f16. Needle hits 500 tokens/sec decode speed on a Raspberry Pi 5, between 400–1,500 tokens/sec on VR devices like Meta Quest 3S and Apple Vision Pro, and ranges 300–700 on sub-$200 phones such as the Samsung A-Series. With a peak session RAM around 28MB, Needle runs on newer microcontrollers like ESP32-S3.

The Playground lets you test Needle for wearables, robots, smart homes, phones, and automotive. Needle is licensed under Apache 2.0, with weights on Hugging Face; the repo gets you running.

45M PARAMS	**800+ tok/s** PI5 PREFILL	**500+ tok/s** PI5 DECODE
CQ2-bit COMPRESSION	**14 MB** FILE SIZE	**28 MB** SESSION RAM

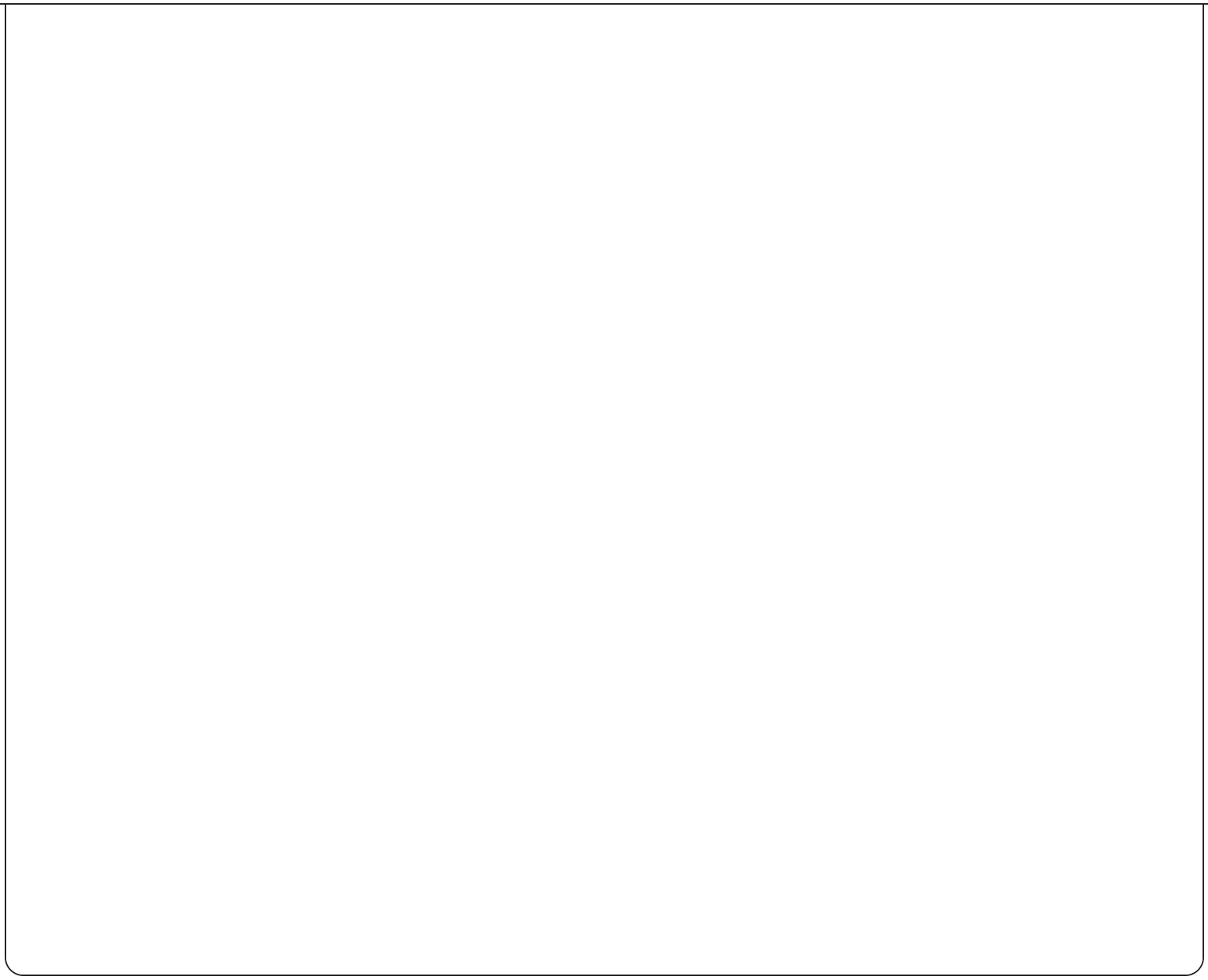

Figure 1. Ordered strict exact match on Mobile-Actions (google/mobile-actions eval split, 961 rows) against total parameters, over the smallest models designed for smart devices, mobile and below. Needle 2 is measured end-to-end through the shipped binary at CQ2-bit deployment precision with tool retrieval on; baselines run the released checkpoints under vLLM, and Apple FM runs on-device.

Our Bet

roughly 1.5 billion PCs, and in emerging markets most phones ship under $200. Count budget phones, Raspberry Pis, microcontrollers, wearables, small robots like Reachy Mini, and connected home devices, and roughly four in five edge devices cost under $200. That is the hardware Needle targets: no GPU, no NPU, a few hundred MB of RAM.

Function Call & Device Use: Turning on a light does not need a frontier model. A watch, a home, a robot: each already exposes its abilities as functions with typed parameters, so the only hard part is mapping a messy sentence onto them: which function, with which values. Framed that way, the problem needs no world knowledge and no open-ended prose, which is why 45M parameters suffice where chat needs billions. That smaller formulation is the bet everything else follows from.

Extraction & Structured Outputs: The schema is the interface, and the same formulation covers documents: a schema plus a paragraph returns typed fields, an enum field is a classifier, an array field collects a list in one call. We enforce this with a contract, not a convention: every turn is answered with a call envelope, the empty call is the refusal, and a byte-level grammar compiled from the declared schemas constrains every token. The grammar carries the syntax, so all 45M parameters go to choosing functions and grounding arguments in the user's words.

Edge-Cloud Collaboration: No small model covers everything, so Needle says so instead of guessing: every response carries a learned confidence score, and off-topic requests return the empty call. Above your threshold, act; below it, re-ask or escalate to the cloud. Most device requests are routine control, so escalation stays rare and the default path stays private, instant, and free.

Lossless 2bit Quantization: Small models break under post-hoc quantization, so we never quantize post-hoc: Needle 2 trains against Cactus Quants from pretrain through post-train, weights, activations, and KV cache alike. The 2bit model you deploy is the model that was trained. That is what fits 45M parameters into 14MB with nothing lost on our battery.

weights: a single dependency-free C++ binary that probes the CPU at startup and picks its kernels, with the model, tokenizer, and grammar compiler sealed inside. One artifact runs from Cortex-M to x86 to WebAssembly. There is nothing to install and nothing to download.

Fine-tune on your Mac/PC: Every product has its own tool vocabulary, and a 45M model is small enough to retrain where it runs: the repo and python package tune and test on your own computer in minutes to a few hours. Ship a Needle that speaks your device's tools, not a generic assistant.

Production

Needle is production-ready for products that require a minimal RAM footprint, low latency, privacy, and offline reliability. Pebble - the pioneer of the modern wearable industry - runs it locally in the Index 01 app to turn spoken requests into actions without depending on a network connection.

> The Pebble Index Ring has no screen. So when you speak to it, the action just has to happen, every time, with or without internet connection. We run Cactus Needle locally in the app, instead of relying on the cloud. The model's footprint is tiny and the performance never lets us down.
>
> **Eric Migicovsky**
> Founder, Pebble

Architecture

The Simple Attention Network

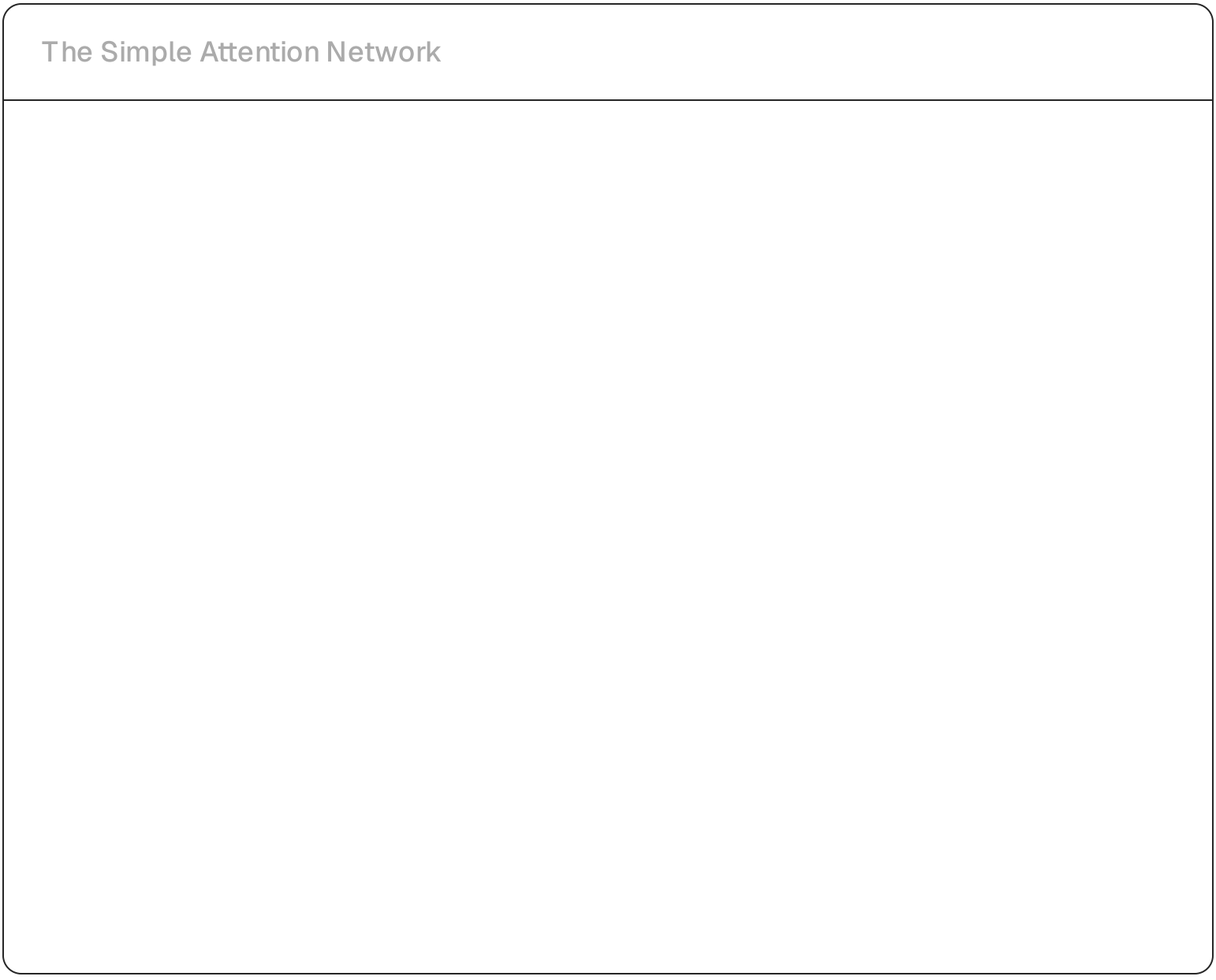

Figure 2. The Simple Attention Network. Each block carries its update rule. Here $\hat{x}$ is the RMS-normalised flattening of the four residual streams, H the orthonormal Walsh-Hadamard transform—a fixed matrix, applied in n log n time with no weights to read—(k_i, v_i) rows gathered from hashed n-gram tables, and P the doubly-stochastic normalisation of the routing logits A, computed by Sinkhorn iteration; a, b, g and all σ-gates are learned and input-dependent. Both attention and MLP residuals are sandwich-normed and gated, the engram sites fire at two layers, and decoding is constrained by a byte-level grammar compiled from the declared schemas.

scale: LFM2.5-230M was pretrained on 19 trillion tokens, roughly 120× Needle's total, and the evaluation below shows the two trading wins. Each component exists to buy capability without buying bandwidth. The Hadamard MLP replaces the usual dense up-and-down projections with a fixed Walsh transform and learned diagonals, so the channel mixing that dominates a small model's weight reads costs almost no parameters at all. The engram moves world knowledge out of the stack into hashed n-gram tables that are read a few rows per token: capacity that is nearly free at decode time, which matters on devices where every megabyte read from flash is latency and battery. The multi-lane residual streams give a 27-layer, 512-wide network the routing flexibility of a much wider one, at the cost of a few dot products per layer rather than more attention or MLP volume.

The memory system is designed backwards from fixed-RAM devices. Attention uses a 256-token sliding window so the KV cache is bounded no matter how long a session runs, and the system prompt and tool declarations are pinned as permanent sinks so the one thing a tool-calling model must never forget—its tools—is structurally unable to be evicted. The cache itself is trained with QAT, and weights are stored in Cactus Quants at a mixed bits per weight averaging 2bit. The result is that quality decisions and deployment decisions stay decoupled: one trained model, specialized to whatever precision and window a target device can afford.

The engine earns its speed from what it refuses to compute. Weights never decompress into RAM: the 2-bit codes are expanded inside vector registers, fused into integer dot products, so resident memory stays at blob size and the arithmetic path is int8 end to end—activations, KV cache, and the lane routing tables alike. The grammar is an optimization, not just a guarantee: because the matcher knows which tokens are legal before the logits exist, the engine computes output scores only for candidate rows, skipping up to 98% of the vocabulary projection on structural tokens, and skips it entirely on steps whose output is already forced. One universal binary probes the CPU at startup and self-selects its kernel tier—SDOT, NEON, AVX2, RISC-V vectors, wasm SIMD, or scalar—and the thread pool spins through the short serial sections of a token

path.

All of it is ultimately an energy argument. On device silicon, moving a byte out of flash or DRAM costs orders of magnitude more than a multiply-accumulate, so the budget that matters is FLOPs per token and bytes per token together. The architecture cuts the first: a conventional transformer of Needle's width and depth spends 164 MFLOPs per token, and even one squeezed down to Needle's parameter count spends 87, because every parameter it owns must be exercised through a matmul. Needle spends 70, and keeps a fifth of its parameters as gathered memory that costs no arithmetic at all. The binary cuts the second, as the engine section showed: nothing rematerializes, the arithmetic stays int8 end to end, and the grammar prunes compute outright, so decoding a token reads at most the 14MB blob once, and on structural tokens meaningfully less. This is what battery life is made of. Even on a high-end phone, an always-on assistant lives inside a power budget; every MFLOP is milliwatt-hours, and Needle spends 7× to 85× fewer of them per token than the models it is benchmarked against.

Compute per token

Model	Params	Matmul-active	MFLOPs / token
Needle 2	45M	35M	70
Same-shape transformer, dense MLP	82M	82M	164
Transformer at matched params	43M	43M	87
LFM2.5 230M	230M	230M	460
FunctionGemma 270M	270M	270M	540
Apple FM	~3B	~3B	~6,000

rows count every parameter as matmul-active, which is exact for both: LFM2.5's eight short-conv blocks hold their parameters in dense gate and projection matmuls that run every token—the depthwise conv kernels themselves are negligible—and what short convolutions save is the context-dependent attention term, already excluded for every row. Tied embeddings count once as the output head. FunctionGemma's 540 is dominated by that head: 170M of its 270M parameters are a 262k-token embedding table.

Bounded session memory is what puts microcontrollers in reach. Because the sliding window caps state, Needle 2's RAM is a deterministic 28MB ceiling, not a curve that grows with conversation length. That fits MCU-class parts with external RAM, such as ESP32-P4 with 32MB of PSRAM, or STM32H7 and NXP i.MX RT boards with SDRAM. The engine compiles single-threaded for bare metal and ships as a static library for Cortex-M4, M7, and M55.

Evaluation

We evaluate on five public function-calling benchmarks: Google's Mobile Actions, DroidCall, the Seal-Tools in-domain and out-of-domain tests, and BFCL v4 single-turn. Scoring is ordered strict exact match: a row passes only if the function names, the call order, and every argument value match. All Needle 2 numbers are measured end-to-end through the shipped C++ engine in its production configuration: CQ2-bit weights, tool retrieval on, and the 256-token sliding KV window. Nothing is relaxed for benchmarking; the numbers reflect the exact engine a device runs, window eviction included. Baselines run the released checkpoints under vLLM at full context, and Apple FM runs on-device.

Two asymmetries make this comparison hard, and we state both upfront. Precision: the baselines stay at f16 deliberately, because conventional post-training quantization to 2

baselines. Scope: Needle is trained specifically for agentic tool calling and nothing else, while every baseline is a general language model carrying chat, prose, and world knowledge alongside its tool calling. That skew favors Needle. There is no clean way to level both at once, so we do not try. The tables answer one narrow question: which model executes tool calls correctly within an on-device budget. We accept the skew; it still paints the picture we intend.

Mobile Actions (961 rows)

Model	Accuracy	Name acc.	Non-empty	1-call	2-call
LFM2.5 230M (f16, vLLM)	69.1	93.0	98.9	76.1	55.0
FunctionGemma 270M (f16, vLLM)	64.0	87.3	98.9	73.0	46.2
Needle 2 (CQ2-bit)	63.7	98.3	99.4	71.3	48.4
Apple FM (on-device)	57.6	94.2	95.5	64.5	43.8

Google Mobile Actions eval split, ordered strict exact match; function names, call order, and every argument must match.

DroidCall test split (200 rows)

Model	Accuracy	Name acc.	Non-empty	1-call	2-call
FunctionGemma 270M (f16, vLLM)	17.5	37.5	59.5	22.7	0.0
Needle 2 (CQ2-bit)	17.0	36.5	47.5	22.1	0.0
LFM2.5 230M (f16, vLLM)	11.0	21.5	22.5	14.3	0.0

Seal-Tools in-domain (700 rows)

Model	Accuracy	Name acc.	1-call	2–3-call	4+-call
Needle 2 (CQ2-bit)	32.6	64.9	63.0	21.8	14.6
LFM2.5 230M (f16, vLLM)	26.9	45.4	54.5	17.1	10.4
FunctionGemma 270M (f16, vLLM)	16.3	56.0	47.0	4.5	2.1

Large candidate tool lists with a majority of multi-call rows.

Seal-Tools out-of-domain (654 rows)

Model	Accuracy	Name acc.	1-call	2–3-call	4+-call
Needle 2 (CQ2-bit)	28.7	58.7	56.4	27.1	15.4
LFM2.5 230M (f16, vLLM)	17.0	35.0	42.6	13.7	9.8
FunctionGemma 270M (f16, vLLM)	15.6	48.9	50.0	11.0	6.3

Entire tool domains are held out of training, testing schema generalization.

Needle was not trained for general function calling: its corpus is consumer device actions—smart home, mobile, wearables, TV, car—plus structured extraction, and BFCL's general-purpose and enterprise API surfaces, including the Java and JavaScript SDK categories, sit entirely outside that distribution. It extrapolates nonetheless: on Python simple calls it lands within a point of FunctionGemma, a model six times larger trained for exactly this task, and it keeps a 93.4 well-formed rate

BFCL v4 single-turn (3,641 rows)

Category	Apple FM on-device	LFM2.5 230M f16 · vLLM	FunctionGemma 270M f16 · vLLM
Simple	**73.3**	63.2	48.1
— Python	**86.8**	85.5	62.3
— Java	**67.0**	48.0	38.0
— JavaScript	**66.0**	56.0	44.0
Multiple	**84.0**	78.5	60.0
Parallel	**65.0**	64.0	36.5
Parallel multiple	**52.0**	51.5	30.5
Live simple	**70.5**	45.0	33.7
Live multiple	45.9	**47.8**	25.2
Live parallel	**50.0**	43.8	18.8
Live parallel multiple	**58.3**	45.8	25.0
Relevance	**100.0**	68.8	81.2
Irrelevance	28.3	**77.7**	72.1
Overall	**61.7**	60.8	46.1
Well-formed rate	95.0	94.2	**100.0**

Official BFCL v4 single-turn scorer. Overall is the collector's unweighted mean over all 13 raw categories; bold values mark the best result in each row.

Building on-device actions into hardware?

Needle is Apache 2.0-licensed. For custom tool schemas, hardware tuning, and post-training, we'd love to talk.

Talk to us Read the docs