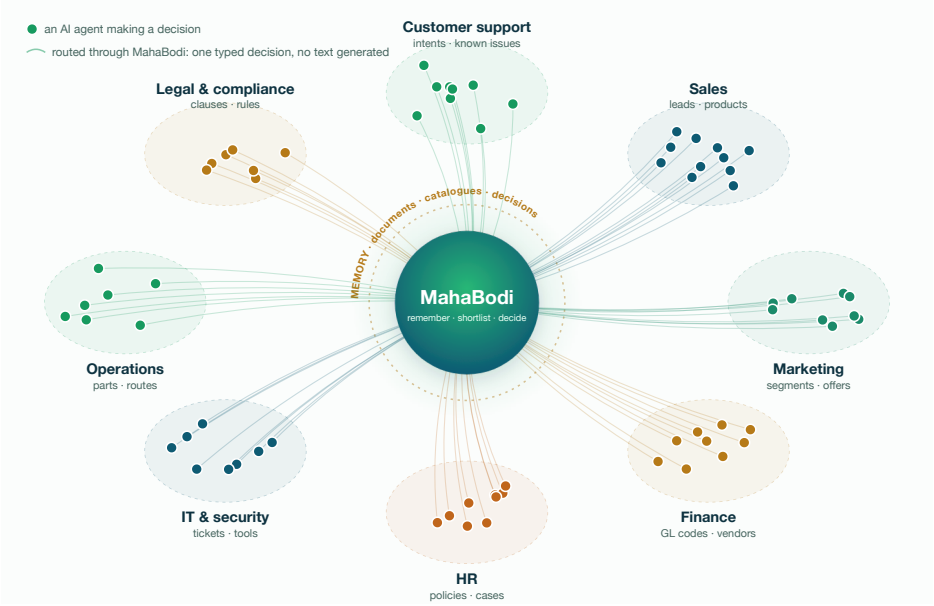
MahaBodi

Memory-driven System-1 decisions for AI agents

mahabodi.github.io
github.com/mahabodi/mahabodi · MIT license

Across business functions, AI agents make many small decisions: which intent, product, entity, tool or policy, out of thousands. MahaBodi is built to route them all: it **remembers** your options, documents and past decisions, **shortlists** the candidates, and **decides** in one forward pass. It returns a typed answer with probabilities and generates no text.



Illustrative uses; measured results are in the table.

HOW IT WORKS

- **Memory:** fastmemory graph memory with typo-tolerant hybrid search; experience memory learns from labelled cases instantly, no retraining.
- **Decisions:** Laya's typed choice, score and yes/no models, native in Rust; tournament shortlisting for many options.
- **Engine:** one Rust core with bindings for Python, Node.js, Java, C#/.NET and Go.
- **Scale path:** memory on PostgreSQL + pgvector (+ Apache AGE); 5.9M Wikipedia pages loaded on one Mac mini, decisions at that scale still being measured.

WHEN TO USE IT

> **2–77 fixed labels, GPU and labelled data:** fine-tune a classifier. A fully fine-tuned Laya beats MahaBodi on 4 of 6 suites.

> **Small label sets, no training step:** MahaBodi (never below Laya as shipped on 5 fresh suites), or a plain kNN, a strong alternative.

> **Thousands to millions of options:** what the memory remembers and retrieves matters more than the decision model. Measured once (entity linking, 10K–100K pages): a remembered prior beat every system, MahaBodi included.

MEASURED, NOT PROMISED

Banking77, 77 intents, zero-shot	0.660 vs Laya 0.492; vs Laya + shortlist 0.642	beat / tie
MASSIVE intent, 51 languages	0.405 vs 0.366	beat
CLINC150, 150 intents, fresh items	0.786 vs 0.756*	beat
BoolQ answered from memory	0.782 vs 0.424 question-only (always-yes 0.626)	beat
Entity linking, 100K candidate pages	0.177 vs 0.121*, via retrieval	beat
Entity linking, 10K candidate pages	0.384 vs 0.387*	tie
Laya fully fine-tuned, 6 small-label suites	fine-tuned Laya wins 4, ties 2	loss
Entity linking vs a remembered alias prior	prior 0.78 beats every system	loss
Latency on 77 options, CPU	730 vs 379 ms	slower

* vs Laya with a MiniLM / dense shortlist, the fair baseline; Laya alone cannot read 10,000 options.

GET STARTED

Python	`pip install "mahabodi[laya]"`
Node.js	`npm install mahabodi`
Rust	`cargo add mahabodi`
.NET	`dotnet add package MahaBodi`

Model weights are not bundled: export them with `research/export_onnx.py`.

Same machine and checkpoint for both systems, seeded samples, exact McNemar on the same items; a beat only at $p < 0.05$. Losses are published next to wins, and every number is recomputed by a second reviewing agent. Full tables: BENCHMARKS.md in the GitHub repo. Alpha, version 0.1.2 (prebuilt for Linux and macOS x86_64).