Explicio

LAB REPORT · EXPLICIO PAIR4

# Small, Calibrated, Verifiable

How Explicio builds its own decision models — a lab report on Explicio pair4




AS OF

As of October 6, 2026, based on measurements from September 28 to October 5, 2026

AUTHOR

Christian Thiede
TACP Operations GmbH & Co. KG (Explicio), Detmold, Germany

# Contents


1 **Executive summary** 3

2 **Why three decisions come before every search** 5

Four ways to make a decision 5

3 **How pair4 came about** 8

Starting point: pair3 8

Method 8

The road to pair4 9

Architecture 12

4 **Results** 14

The three decisions compared 14

Where others do better 15

Speed and location 15

5 **Operation and trust** 16

Operation 16

Trust 16

6 **Limits and outlook** 18

Limits 18

What comes next: Explicio pair5 19

Five questions for any vendor 19

7 **Provenance and reproducibility** 20

About this document 20

CHAPTER 1

# Executive summary



Before a knowledge assistant searches, it needs to know three things. What is the message – a question, a piece of knowledge, a reply to a clarifying question? How is a short reply such as "fine by me" to a clarifying question meant? And does the new message need the conversation so far for the search to find the right thing? Get these three decisions wrong and the assistant searches with the wrong question, files a question as knowledge, or links things that do not belong together. Answer quality starts before the search.

Explicio makes these decisions with a small model of its own. Its current generation is called **Explicio pair4**. It is built on an openly licensed multilingual encoder with 307 million parameters, trained only on data we generated ourselves, and it runs on the CPU of the Explicio Box, without a graphics processor. We measured it on test sets that every model sees exactly once: against the large language model that writes the answers on our Boxes (Qwen3.6-35B-A3B), against a model from another vendor fine-tuned for decisions (Nimble 9B by Bespoke Labs), and against our previous model, pair3.

On what a message is asking for, pair4 reaches 95.5% across 19 languages, the large language model 83.3%, Nimble 72.5%. On whether a message needs the conversation so far, pair4 reaches 96.4% balanced accuracy on a new test set, its predecessor 93.8%. One decision takes 62.8 ms on the CPU of the Box. We do not lead everywhere: on an older test set, Nimble gets follow-up detection right more often than pair4. The gap is within the noise, but it does not favour us, and this document reports it next to the results.

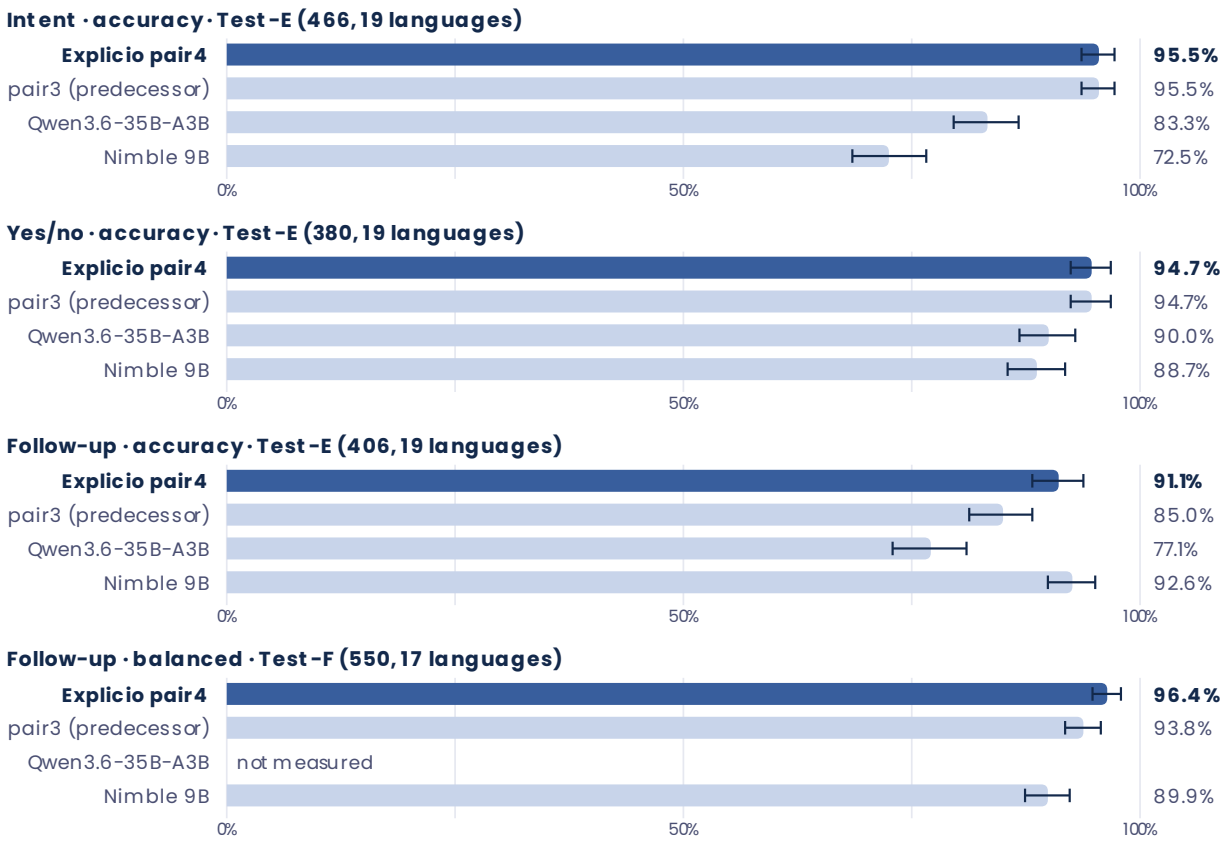
The technical point of this report is an insight that cost us three discarded attempts. The encoder was not short of knowledge. The bottleneck was the average over every word that the old output head used to read that knowledge out. An attention head that reads the words themselves closes most of the gap without touching the encoder – and therefore without changing the other two decisions.

| Metric | Value | Source |
|---|---|---|
| Intent, accuracy, Test-E (466 messages, 19 languages) | pair4 95.5% · Qwen3.6 83.3% · Nimble 72.5% | `teste-pair4.ndjson`, E4 raw data `qwen36-frei.ndjson`, `nimble-info.ndjson` |
| Yes/no, accuracy, Test-E (380 replies, 19 languages) | pair4 94.7% · Qwen3.6 90.0% · Nimble 88.7% | same files |

| Metric | Value | Source |
|---|---|---|
| Follow-up, accuracy, Test-E (406 messages, 19 languages) | pair4 91.1% · Nimble 92.6% · pair3 85.0% · Qwen3.6 77.1% | same files, `s4b/pair3-teste.ndjson` |
| Follow-up, balanced accuracy, Test-F (550 messages, 17 languages) | pair4 96.4% · pair3 93.8% · Nimble 89.9% | `f2e/onnx/testf.ndjson`, `f2a/rohdaten/f2a-nimble-testf.ndjson` |
| Time per decision, Box CPU, p95 | 62.8 ms (all three in one call: 202.4 ms) | `f2e/box/bench-box-pair4.json` |
| Follow-up: statistically proven error bound at 95% confidence | 4.8% at 95.3% coverage | `f2e/gate/gate.json` |
| Intent and yes/no compared with pair3 | bit-identical on 3,700 cases (Δp = 0) | F2e report, rule E2 |
| Memory of the loaded model | 1,526 MiB | `f2e/box/bench-box-pair4.json` |

*Every figure is listed with source, sample size, date of measurement and 95% interval in the number sheet `docs/publikationen/pair4/zahlenblatt.json`, which a script counts from the measurement rows (`tools/publikationen/pair4_zahlen.py`). All test sets are synthetic.*

**FIG. 1** **Three decisions on the release sets: bars start at 0, whisker = 95% interval, all sets synthetic**




Bars start at 0, whisker = 95% interval (bootstrap, 10,000 resamples). All sets synthetic.

CHAPTER 2

# Why three decisions come before every search



Explicio collects knowledge where it arises: in chats, voice messages, photos and documents. It asks when something is missing and answers questions with evidence from a knowledge graph. Along the way, every message triggers a series of small decisions, long before a language model phrases an answer. pair4 makes three of them today. The examples in this section are made up; we do not quote items from the test sets.

**Intent.** "Who looks after the chiller in hall 3?" is a question. "From now on, team North services the chiller in hall 3." is a piece of knowledge. Then there is small talk, getting-started questions, replies to an open clarifying question, and requests to assess a person – which Explicio recognises and declines. Get this wrong and a question ends up in the graph as knowledge, or a contribution gets answered as if it were a question.

**Yes/no.** Explicio asks back, for example: "Do these two entries belong together?" The reply rarely says "yes". It says "fine by me", "not really, that's the old unit" or "no idea". How it is meant decides whether two pieces of knowledge get linked. A word list that looks for "yes" and "no" gets 34.5% of our test set right.

**Follow-up.** After "What is the test pressure of pump P-12?" comes "And for the P-14?". On its own, that message cannot be searched; it needs the conversation and is rewritten before the search. "How long is the induction for new colleagues?", on the other hand, stands on its own, and rewriting it would mix it up with pumps. A missed rewrite costs the answer; an unnecessary one costs a call and, at worst, precision. The rule that decides in production so far – lists of pronouns and pointing words plus length and first word – reaches a balanced accuracy of 57.6% on our calibration sets.



IN SHORT

Part of an answer's quality is decided before the search. Three small decisions determine what is searched for and what enters the knowledge graph. They have to be fast, multilingual and verifiable.

## Four ways to make a decision

There are four common approaches to this kind of decision today. We built or measured all four. The overview below is a guide; the figures come from our measurements on our test sets and hold for those.

### Rules and word lists

A list of words and patterns, maintained per language. This is where Explicio started, and a word list still sits in front for the unambiguous cases.

| Advantages | Drawbacks |
|---|---|
| instant, no compute | on our sets 34.5% (yes/no) and 57.6% balanced (follow-up) |
| fully traceable | needs lists per language; languages that drop pronouns slip through |
| no training data | knows no context: "and …" always counts as a follow-up |

### A large language model with instructions

A large language model receives the message plus a description of the classes and picks one. We measured Qwen3.6-35B-A3B, the model that writes the answers on our Boxes, as a classifier with exactly one output token: one letter per class, the most likely one wins.

| Advantages | Drawbacks |
|---|---|
| no training data; a new class is a new line in the description | without training below our small model in all three decisions: 83.3% / 90.0% / 77.1% on Test-E |
| runs on existing hardware, no new service | needs the GPU that also writes the answers; 151 ms per decision (p95, over the network) |
| understands many languages | less well calibrated: its confidence says little about how often it is right (calibration error up to 0.144 against at most 0.048 for pair4) |

### A specialised language model from another vendor

Vendors now offer small language models fine-tuned for exactly this kind of decision. We measured Nimble 9B by Bespoke Labs (Apache-2.0) and wanted to measure Clef Flash by Cloudflare.

| Advantages | Drawbacks |
|---|---|
| strong on follow-up detection without our own training: 92.6% on Test-E | takes about 16 GB under Ollama and cannot run on the Box next to the answer model; Clef Flash failed while loading |
| open licence (Apache-2.0), published training data | 524 ms per decision (p95, GPU, no network hop) |
| | weak on intent (72.5%), behind our predecessor on the new set Test-F |

### A small encoder, trained in-house

An encoder reads message and context together and outputs probabilities for each decision. This is the route of pair3 and pair4.

| Advantages | Drawbacks |
| --- | --- |
| on the CPU: 62.8 ms per decision, 1.5 GiB of memory, the GPU stays free | needs training data; every new task is a new training run |
| calibrated: its confidence carries a statistically proven error bound | the data is synthetic; how well it matches real traffic only production will show |
| one model, one pin, deterministic on the CPU | knows exactly its tasks and nothing beyond |



**IN SHORT**

For three fixed, frequent decisions, a small encoder trained for the job is more accurate, faster and better safeguarded on our sets than a large language model. The large model keeps the advantage where the task keeps changing.

CHAPTER 3

# How pair4 came about



## Starting point: pair3

pair3 is a sentence-pair encoder on `jhu-clsp/mmBERT-base`, a multilingual encoder under the MIT licence: 22 layers, 307 million parameters, 197 million of them in the embedding of a vocabulary of 256,000 tokens. The input is one sequence of context and message – `[CLS] context [SEP] message [SEP]`, at most 320 tokens – so every layer computes attention across both parts. On top of the last state sits one linear head per decision, fed by the average over all tokens.

pair3 was trained three times independently with different seeds on the GPUs of the Boxes; what ships is the arithmetic mean of the weights. Averaging measurably helps: the single runs reach 93.0 to 93.5% on intent on the calibration sets, the average 94.2%. Each decision gets a temperature, and yes/no and follow-up get a gate: a threshold above which the model decides on its own, chosen so that the error rate stays below a fixed limit with 95% confidence.

The training data comes in two parts. The first was written with Claude, the second generated by the answer model on our own Boxes through the Explicio Grid, with the class set in advance by a slot list and checked blind by Claude subagents. Together that is 2,423 examples for intent, 3,137 for yes/no and 1,820 for follow-up, in 18 languages, all cleaned against every test set.

pair3 had one clear weakness. Messages on the same topic that still make sense on their own – a change of subject starting with "and", a pronoun resolved within the same message, polite forms, bare keywords – it often took as dependent. On Test-E it recognised only 64.9% of the stand-alone messages. The ranking was right (ROC-AUC 0.979); the boundary sat too far on the "dependent" side.

## Method

### A specification before the build

Before pair4 was built, we wrote down what it had to deliver: a specification with must and should values for every decision, for languages, calibration, speed, memory, licence and data. The key must values: intent and yes/no no worse than pair3; follow-up on the new release set Test-F with at least 30% fewer balanced-accuracy errors, so at least 95.625%; the should value was 40% fewer errors, 96.25%. Dependent messages at least 98%. And pair3's operating envelope: one model, one pin, CPU, at most 1.6 GiB.

We moved two limits along the way, and both are reported here. The Test-F threshold was originally 40% fewer errors; it was lowered to 30% before the measurement, because even the projection for a fully trained model sat within the noise of the old threshold. The latency limit on the Box was 200 ms for a call with three decisions; pair4 came in at 202.4 ms, and the limit was set to 210 ms **after** the measurement. The operational requirement for this location was 300 ms; the 200 ms had been set as pair3's envelope with a 3% margin.

### Rules fixed before the measurement

**Criteria first.** For every session, thresholds and the analysis script are in version control before the first number exists. A rule may change after the result if the error demonstrably lies in the rule – as an addendum with a reason – and no gate counts as passed because of it.

**Release sets once.** Every model sees Test-E and Test-F exactly once, by explicit name. They are never part of training, calibration or the gate, and nobody looks at individual items to tune anything. Selection happens on the calibration sets B, C and D.

**Synthetic and checked blind.** Test-E was written by Claude subagents (Claude Sonnet) following rules and checked blind by Claude Opus. Test-F was written by twelve Claude Opus instances, one industry each and at most three languages; six Claude Sonnet instances checked it blind and agreed in 574 of 576 cases. One author instance that had opened another's file was discarded and replaced.

**Paired, with intervals.** We compute differences between two models paired, item by item, with a 95% interval over 10,000 bootstrap resamples; for balanced accuracy stratified by class.

**Measured as it runs.** pair4 is measured as the same ONNX graph that runs in the cloud and on the Box. Timing comes from a Box, in the same run as pair3.

**At least three runs.** Every arm is trained at least three times with different seeds, the average ships, and every change runs against a control of the same size and step count.

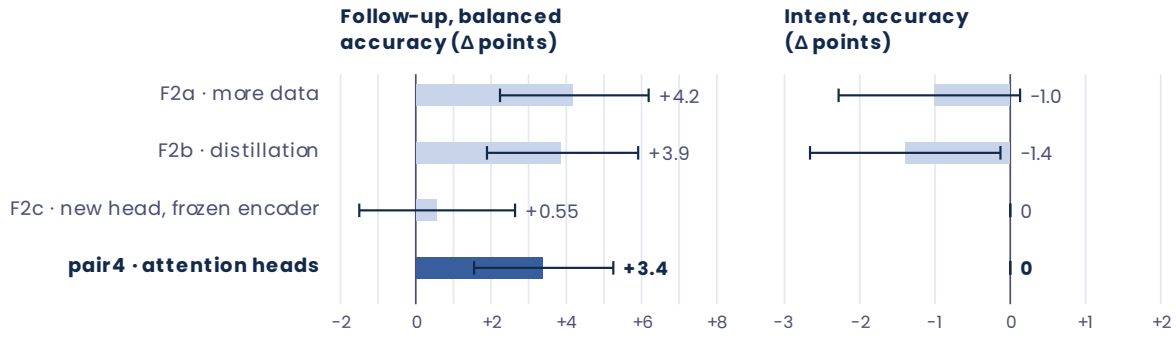
## The road to pair4

From the first attempt on September 24 to the acceptance of pair4 on October 5:

| Date | Step |
|---|---|
| September 24/25 | First attempts on an off-the-shelf multilingual classifier; its lineage includes a non-commercial dataset. Restart on mmBERT-base (MIT) |
| September 26 | pair1, the first sentence-pair model of our own, in shadow mode |
| September 27 | pair2: the average of three runs instead of a single one |
| September 28 | pair3 passes the release set Test-E; since then the reading of yes/no replies takes effect as a question back to the person |

| Date | Step |
|---|---|
| October 1 | pair3 runs on the CPU of the Boxes; a smaller base model is measured and dropped (intent -8.2 points) |
| October 4/5 | off-the-shelf decision models measured: Qwen3.6 and Nimble; Clef Flash could not be measured |
| October 5 | F2a to F2c dropped, F2d finds the bottleneck, F2e builds pair4; accepted and deployed in the cloud and on the first Box |

**FIG. 2** **Gain on follow-up, loss on intent**




Paired against pair3 on the calibration sets B+C+D (589 and 790 cases), whisker = 95% interval.

All values paired against pair3 on the calibration sets B+C+D, follow-up as balanced accuracy, intent as accuracy. Source: `weg_bcd` in the number sheet.

## F2a: more data, with a teacher

The weakness on stand-alone messages looked like a data gap. We had the answer model on our Boxes write 1,667 new follow-up candidates and put Nimble on top as a teacher: it confirmed 98% of the dependent candidates but only 38% of the stand-alone ones – and was mostly right in our spot check. 963 examples remained. Training further on them raised follow-up by 4.2 points. Intent dropped by 1.0 points. One encoder for three tasks: what helps one shifts the others. Under the rules fixed in advance ("nothing else gets worse"), no switch.

## F2b: holding on to the predecessor

If the encoder changes, intent and yes/no could learn from the predecessor how to answer – distillation with pair3 as the teacher. Follow-up rose by 3.9 points, intent still dropped by 1.4. The reason is in the teacher's numbers: on its own training data, pair3 is almost always sure (median 0.996). There are hardly any soft targets a student could learn from. No switch.

### F2c: freezing the encoder

If the encoder must not change, only the head is left. We retrained the linear follow-up head on top of the frozen pair3, with all the data. Intent and yes/no stayed bit-identical, the same probabilities on 3,700 cases. But follow-up gained only +0.55 points, and ROC-AUC dropped: the new head shifted the boundary but did not separate the hard cases any better. No switch.

### F2d: the insight

Three attempts, one pattern. Where the encoder learns, follow-up gains and intent loses. Where it stays frozen, nobody loses, but nobody gains either. The obvious reading would be that the 110 million encoder parameters beyond the embedding are not enough for three tasks.

The measurements say otherwise. With a learning encoder, the same linear head reaches a balanced accuracy of 94.5% – so the relations between history and message can be represented. And pair3 is a cross-encoder: each of its 22 layers computes attention over history and message at once. What F2c shows is narrower: the **frozen** encoder with an **average** over all tokens and a linear head is not enough. The average is the suspect. It folds twenty or two hundred token states into one vector and in doing so blurs which word refers to what. Whether "it" in the new message points to something in the history or to something in the same message is visible in the states of the individual tokens – after averaging, it is not.

So we replaced the average, not the encoder. The new head is a small attention head with 0.89 million parameters: the tokens of the new message query the tokens of the history, over the unchanged states of the frozen pair3. Result on the calibration sets: balanced accuracy 93.96% instead of 90.61%, ROC-AUC 0.985 instead of 0.979. The head closes 83% of the gap between the linear head from F2c and the fully trained encoder from F2a – without changing a single encoder parameter.



**IN SHORT**

The encoder was not short of knowledge. The bottleneck was how it was read out. A head that reads the words themselves gets out of the same representation what the average blurs – and leaves everything else untouched.

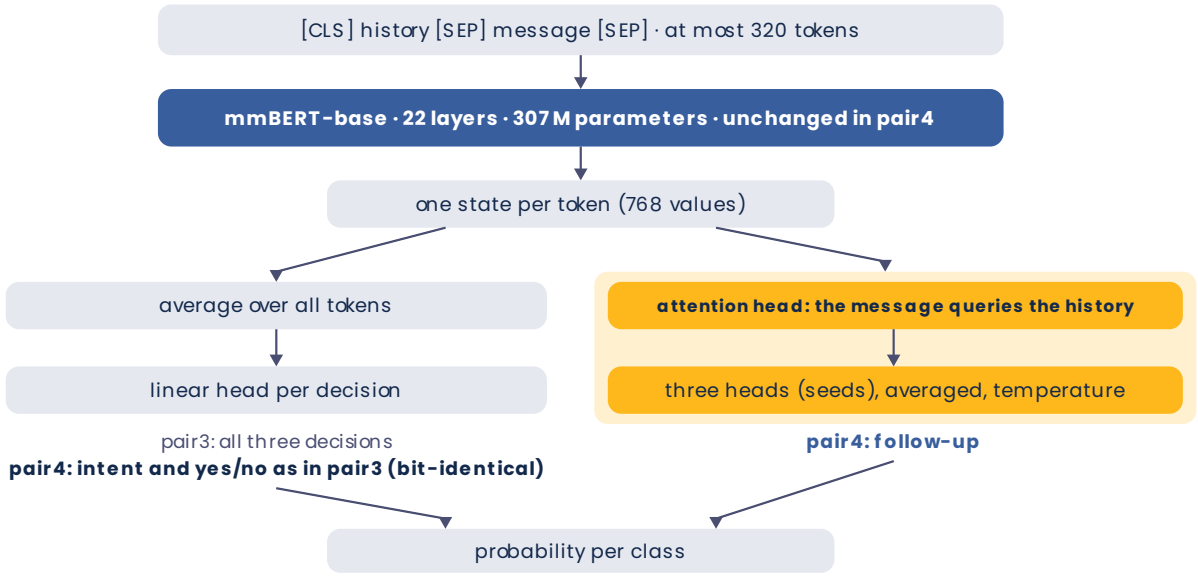
### F2e: pair4

The pilot became a model. Three such heads, trained with different seeds, sit inside pair4's ONNX graph; their probabilities are averaged and calibrated with a shared temperature of 0.88. Intent and yes/no run unchanged through pair3's heads.

Acceptance followed the specification. Export parity: 774 of 774 cases with the same choice, deviation at most $3 \cdot 10^{-6}$. Intent and yes/no bit-identical to pair3 on 3,700 cases. Follow-up on Test-F: balanced 96.42% – the should value of 96.25% is met. Dependent messages 99.63%, one error in 270. Calibration error (ECE) 0.019. The only line missed was the latency limit on the Box (202.4 instead of 200 ms); the method section describes it.

## Architecture

**FIG. 3** **From the average to an attention head**

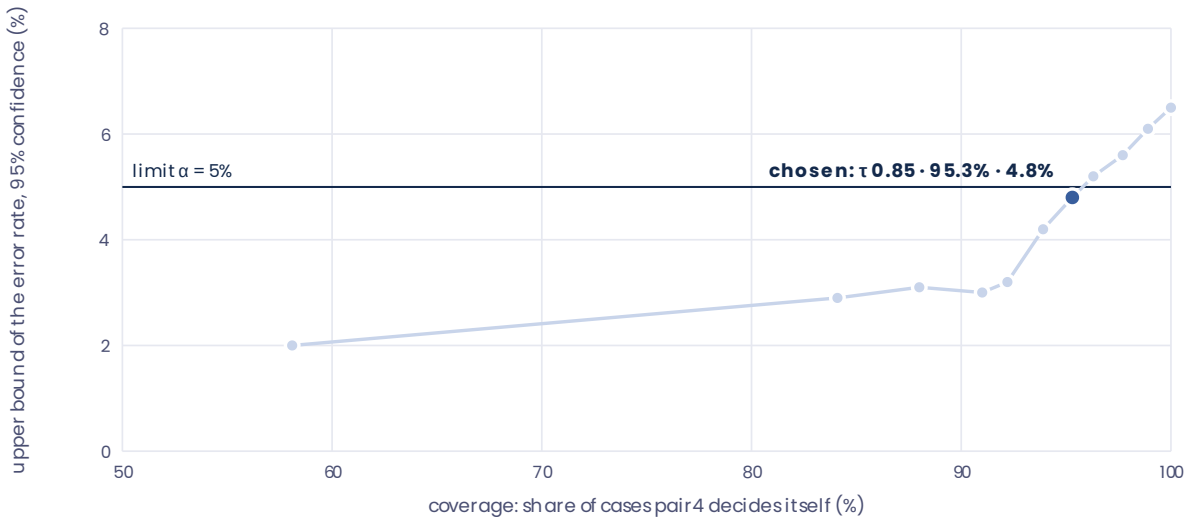



One ONNX graph, same inputs and outputs as pair3. The average blurs which word refers to what; the head reads the tokens themselves.

pair4 is a single ONNX graph with the same inputs and outputs as pair3. The runtime in the `ai-gateway` did not need to know anything had changed: for follow-up rows it reads a different output. The masks that separate history and message inside the head are built in the graph from the positions of the separator tokens. The head runs for every row of a call, including intent and yes/no rows whose result the runtime discards. That costs about 3% of the time and explains the 202.4 ms.

The follow-up gate has been recomputed. At a threshold of 0.85, pair4 decides 95.3% of cases on its own; the observed error rate there is 3.0%, the upper bound at 95% confidence 4.8%, below the required limit of 5%. For pair3 the threshold was 0.93, covering 81.3%. The basis is the calibration slice and sets A to D, all synthetic; the gate is therefore provisional until labelled traffic provides the final threshold.

**FIG. 4** **Coverage against statistically proven error bound**




One point per threshold τ (0.50 to 0.99; higher thresholds cover under 5%). Basis: calibration slice and Test-A–D, synthetic, provisional.

One point per threshold τ. Source: `gate_pair4.rewrite.kurve` in the number sheet, from `f2e/gate/gate.json`.

CHAPTER 4

# Results



## The three decisions compared

Figure 1 in the executive summary shows the values as bars; the table gives both measures.

| Decision, set, measure | pair4 | pair3 | Qwen3.6-35B-A3B | Nimble 9B |
|---|---|---|---|---|
| Intent, Test-E, accuracy | **95.5** [93.6; 97.2] | 95.5 | 83.3 [79.6; 86.7] | 72.5 [68.5; 76.6] |
| Intent, Test-E, balanced | **95.4** | 95.4 | 85.8 | 83.4 |
| Yes/no, Test-E, accuracy | **94.7** [92.4; 96.8] | 94.7 | 90.0 [86.8; 92.9] | 88.7 [85.5; 91.8] |
| Yes/no, Test-E, balanced | **94.7** | 94.7 | 90.0 | 88.6 |
| Follow-up, Test-E, accuracy | 91.1 [88.2; 93.8] | 85.0 | 77.1 [72.9; 81.0] | **92.6** [89.9; 95.1] |
| Follow-up, Test-E, balanced | 89.7 | 82.5 | 73.3 | **91.5** |
| Follow-up, Test-F, balanced | **96.4** [94.8; 97.9] | 93.8 [91.8; 95.7] | not measured | 89.9 [87.4; 92.3] |
| Follow-up, Test-F, accuracy | **96.4** [94.7; 97.8] | 93.6 [91.5; 95.6] | not measured | 89.8 [87.3; 92.2] |

*Values in percent, 95% interval in brackets. Source: `zellen` in the number sheet. Switching to the other measure does not change the order in any row.*

Paired, pair4 leads Qwen3.6 on intent by 12.2 points (interval 8.6 to 15.9) and Nimble by 23.0 points (18.7 to 27.3); on yes/no it leads Qwen3.6 by 4.7 points (1.6 to 7.9) and Nimble by 6.1 (2.9 to 9.5). On follow-up detection on Test-F, pair4 leads Nimble by 6.5 points (4.0 to 9.1) and pair3 by 2.7 points (1.4 to 4.1, balanced).

Nimble's intent result shows a striking gap between accuracy (72.5%) and balanced accuracy (83.4%): it mainly misses the large class of requests to assess a person, which it often takes for replies to a clarifying question; it does much better on smaller classes. That does not change the order of the models.

The differences sit exactly where pair3 was weak. On Test-F, pair4 recognises 93.2% of stand-alone messages, pair3 87.5%, Nimble 82.9%. On dependent messages, pair4 misses one in 270, pair3 none, Nimble eight.

## Where others do better

**Nimble on follow-up detection on Test-E.** Nimble gets 92.6% there, pair4 91.1%. Paired, the gap is -1.5 points with an interval of -4.9 to +2.0: within the noise, but not in our favour. On the newer set Test-F the picture reverses. That Nimble does better on Test-E and worse on Test-F points to a dependence on the style of the test sets (Test-E written by Claude Sonnet, Test-F by Claude Opus). A single set cannot carry a claim of "better".

**New tasks.** A large language model picks up a new category from a description. pair4 knows exactly its three tasks; a fourth needs data and a training run.

**The GPU.** On the Box's GPU, a model of this design made all three decisions in 10 ms (p95), about twenty times faster than on the CPU. We run on the CPU because the GPU writes the answers and one memory pool serves both.

**The predecessor.** pair3 recognised all 270 dependent messages on Test-F and was slightly better calibrated there (ECE 0.012 against 0.019).

## Speed and location

| Model | Runs on | p50 | p95 | Cases |
|---|---|---|---|---|
| pair4, one decision per call | Box CPU, 2 threads | 39.3 ms | 62.8 ms | 665 |
| pair4, three decisions in one call | Box CPU, 2 threads | 131.5 ms | 202.4 ms | 221 |
| pair3, one decision per call | Box CPU, 2 threads | 38.1 ms | 60.9 ms | 665 |
| Qwen3.6-35B-A3B, one decision per call | GPU of the Boxes, round trip over the network | 133.0 ms | 151.0 ms | 1,252 |
| Nimble 9B, one decision per call | Box GPU, no network hop, loading excluded | 292.0 ms | 523.5 ms | 1,042 |

*Source: `tempo_box_cpu` and `tempo_sprachmodelle` in the number sheet.*

The language models' timings come from the same runs as their accuracy figures, on the same three tasks of Test-E. The conditions are not the same: the language models run on a GPU, Qwen3.6 including the network path, Nimble without it; pair4 runs on the CPU, and its timing uses 225 reference cases rather than the release sets. The table shows what each model costs where it would actually run. pair4 takes 1,526 MiB of memory and no graphics memory.

CHAPTER 5

# Operation and trust



## Operation

pair4 runs in the `ai-gateway` on the CPU, in the cloud and on the Box, with ONNX Runtime and two threads. On the Box it ships as its own image with the signed release, mounted read-only, and reaches every Box through the update rings. It has been running in the cloud and on the first Box since October 5, 2026.

So far, only yes/no has an effect in production, and only as a question back: if the word list recognises nothing and the model is confident above its threshold, Explicio asks again with its reading, and the effect only follows the person's explicit reply. An effect without that question is possible per tenant only after a separate statistical gate; no tenant has been switched so far. Intent and follow-up detection run in shadow mode: their decisions are recorded and compared with the current path, but not applied. Whether and when follow-up detection takes effect is a separate decision with its own gate.

Because intent and yes/no are bit-identical to pair3, the switch changed nothing that has an effect today. Rolling back is a pointer and a restart; pair3's folder stays in place.

## Trust

**No customer data.** pair4 was trained only on data we generated ourselves: written with Claude, generated by the answer model on our own Boxes, filtered by Nimble, checked blind and cleaned against every test set. No customer knowledge went into any model. Explicio does not learn across tenants; one customer's data never improves a model that another customer uses. Whether data generated with Claude may be used to train a decision model inside our own product was clarified in advance: such a component is not a competing product.

**Licences.** The base model mmBERT-base is MIT-licensed; its lineage names no non-commercial dataset. We dropped a first candidate on September 25 because its training data included a non-commercial dataset. We checked every comparison model before measuring it; Nimble and Clef are Apache-2.0, and another model without a stated licence for its weights did not take part. The licence texts shipped with the Box images are the subject of an ongoing licence package.

**Classification under the EU AI Act.** pair4 sorts messages. It does not assess, select or decide about people; it recognises requests to assess a person as such, and Explicio declines them. Explicio remains a knowledge-management tool; pair4 does not change its classification, and every reassessment is recorded in the company's AI system register. Every action with an effect stays with a person.

**Traceability.** Every shipped model carries a manifest with checksums and is fixed in the code by a pin. Seeds, training plans, criteria files with their commits and the measurement rows are in the repository; every run can be repeated with the same data, seeds and scripts.

CHAPTER 6

# Limits and outlook



## Limits

- **Synthetic sets.** All training and test sets were written by models. The reviewers almost always agreed (574 of 576 on Test-F), so the sets lean towards clear-cut cases; real traffic is fuzzier. How well pair4 does there, only production will show.
- **One set cannot carry a comparison.** Nimble is better than pair4 on Test-E and worse on Test-F. The claim "pair4 beats Nimble" holds for intent, for yes/no and for Test-F, not for follow-up detection on Test-E.
- **The language model as a one-token classifier.** Qwen3.6 was measured with one output token per decision, not with the longer instructions Explicio uses in the product for other tasks. Longer instructions might score better and would be slower.
- **Clef Flash not measured.** Under Ollama, the Box's free memory fell below our safety threshold while the model was still loading. It could be measured with more memory or through another runtime.
- **Timing under different conditions.** The timings of the language models and of pair4 come from different compute paths and case sets; the table compares what each model costs in its place, not a race under equal conditions.
- **One Box, one measurement.** pair4's timing was measured on one Box at one point in time. The latency limit was moved after the measurement.
- **Provisional gate.** The statistically proven bound holds for the synthetic calibration sets. The final threshold will come from labelled traffic.
- **Shadow mode.** Follow-up detection does not take effect yet. The figures describe what pair4 would decide.

## What comes next: Explicio pair5

pair5 is in development, and its targets are fixed before it is built: 95% per decision on a new, independent release set, at least 93% in every major language with a spread of at most 5 points, and all decisions for a message in 40 ms on the CPU of the Box. The routes there are described in our research programme: adapters that act for one decision only; a cascade that settles clear-cut cases in microseconds and passes only the hard ones to the encoder; and, in the longer run, a base model of our own, pre-trained on text from an openly licensed model. It will be measured under the same rules: criteria first, release sets once, and the comparison says where others do better.

## Five questions for any vendor

Anyone choosing a knowledge assistant can put these questions to any vendor, including us.

1. **Which decisions are made before the search, and who makes them?**
   A rule, a large language model, a model of their own?

2. **Which sets were used for measurement, and how often has the model seen them?**
   A figure on a set that was read during development says little.

3. **Is there a statistically proven error bound, and what happens below it?**
   A model that does not know its uncertainty cannot hand it over.

4. **Where does the decision run, and what does it cost?**
   On which hardware, with what latency, next to which other workloads?

5. **Where does another model do better?**
   A comparison that does not say so is an advertisement.

CHAPTER 7

# Provenance and reproducibility



| Component | Version |
|---|---|
| Model | `explicio-dm-pair4`, manifest sha256 `dbf3b079…`, pin `services/ai-gateway/app/decide/pin.json` |
| Base model | `jhu-clsp/mmBERT-base`, revision `c5955035…`, MIT |
| Follow-up heads | three heads, seeds 20261024–26, `koepfe.json` with checksums |
| Runtime | ONNX Runtime 1.28 on the CPU, 2 threads |
| Comparison models | `qwen3.6-35b-a3b-nvfp4` (Qwen3.6-35B-A3B in NVFP4 quantisation) via vLLM; Nimble 9B (Ollama tag `nimble:9b`, LoRA on Qwen3.5-9B) via Ollama 0.35.1; measured on 4 and 5 October 2026 |
| Specification | `docs/lastenheft-entscheidungsmodell.md` |
| Criteria and reports | `docs/messungen/entscheidungsmodelle/2026-10-04-e4-*`, `2026-10-05-f2a-*` to `2026-10-05-f2e-*`, each `-kriterien.md` committed before measuring |
| Measurement rows | `f2e/onnx/`, `f2e/box/`, `f2e/gate/`, `f2a/rohdaten/`, `2026-10-04-e4-rohdaten/`, `s4b/` under `docs/messungen/entscheidungsmodelle/` |
| Number sheet | `docs/publikationen/pair4/zahlenblatt.json`, generated by `tools/publikationen/pair4_zahlen.py` |
| Figures | `docs/publikationen/pair4/figures/`, generated by `tools/publikationen/pair4_figures.py` |

We do not release the release sets themselves, so that they stay unused for future models. Measurement rows with ID, class, prediction and probability, the number sheet and the tools are available to auditors, customers and universities on request.

## About this document

Author: Christian Thiede, TACP Operations GmbH & Co. KG (Explicio), Detmold, Germany, ORCID 0009-0002-4517-7208. An AI assistant (Claude, Anthropic) contributed to development, measurement, analysis and text; the author is responsible for content and figures. Statements about third-party models (Qwen3.6, Nimble, Clef Flash) refer to the versions named, our test sets and the date of measurement. This document may be passed on unchanged.



Explicio®

Das digitale Gedächtnis Ihres Unternehmens

TACP Operations GmbH & Co. KG (Explicio) · Detmold