Explicio

WERKSTATTBERICHT · EXPLICIO PAIR4

Klein, kalibriert, nachprüfbar

Wie Explicio seine Entscheidungsmodelle selbst entwickelt — Werkstattbericht zu Explicio pair4



STAND

Stand 6. Oktober 2026, nach Messungen vom 28. September bis 5. Oktober 2026

AUTOR

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

Inhalt

1 **Zusammenfassung für Entscheider** 3

2 **Warum vor jeder Suche drei Entscheidungen stehen** 6

Vier Wege zu einer Entscheidung 7

3 **Wie pair4 entstand** 9

Ausgangspunkt: pair3 9

Methode 10

Der Weg zu pair4 11

Architektur 13

4 **Ergebnisse** 15

Die drei Entscheidungen im Vergleich 15

Wo andere besser sind 16

Tempo und Rechenort 16

5 **Betrieb und Vertrauen** 17

Betrieb 17

Vertrauen 17

6 **Grenzen und Ausblick** 19

Grenzen 19

Was als Nächstes kommt: Explicio pair5 20

Fünf Fragen an jeden Anbieter 20

7 **Herkunft und Reproduzierbarkeit** 21

Über dieses Dokument 22

KAPITEL 1

Zusammenfassung für Entscheider



Bevor ein Wissensassistent sucht, muss er drei Dinge wissen: Was will die Nachricht – ist sie eine Frage, ein Wissensbeitrag, eine Antwort auf eine Rückfrage? Wie ist eine kurze Antwort wie „passt schon“ auf eine Rückfrage gemeint? Und braucht die neue Nachricht den bisherigen Verlauf, damit die Suche das Richtige findet? Wer diese drei Entscheidungen falsch trifft, sucht mit der falschen Frage, legt eine Frage als Wissen ab oder verknüpft, was nicht zusammengehört. Die Qualität der Antwort beginnt also vor der Suche.

Explicio trifft diese Entscheidungen mit einem eigenen, kleinen Modell. Seine aktuelle Generation heißt **Explicio pair4**. Es basiert auf einem frei lizenzierten mehrsprachigen Encoder mit 307 Millionen Parametern, ist ausschließlich mit selbst erzeugten Daten trainiert und rechnet auf der CPU der Explicio Box, ohne Grafikprozessor. Gemessen haben wir es auf Testsätzen, die jedes Modell genau einmal sieht: gegen das große Sprachmodell, das auf unseren Boxen die Antworten schreibt (Qwen3.6-35B-A3B), gegen ein für Entscheidungen feinabgestimmtes Modell eines Drittanbieters (Nimble 9B von Bespoke Labs) und gegen unseren Vorgänger pair3.

Bei der Absicht einer Nachricht liegt pair4 in 19 Sprachen bei 95,5 %, das große Sprachmodell bei 83,3 %, Nimble bei 72,5 %. Bei der Frage, ob eine Nachricht den Verlauf braucht, erreicht pair4 auf einem neuen Testsatz 96,4 % ausgewogene Trefferquote, der Vorgänger 93,8 %. Eine Entscheidung kostet auf der CPU der Box 62,8 ms. Nicht überall liegen wir vorn: Auf einem älteren Testsatz trifft Nimble die Folgefrage öfter als pair4; der Abstand liegt im Rauschen, spricht aber nicht für uns. Dieses Dokument nennt das an derselben Stelle wie die Ergebnisse.

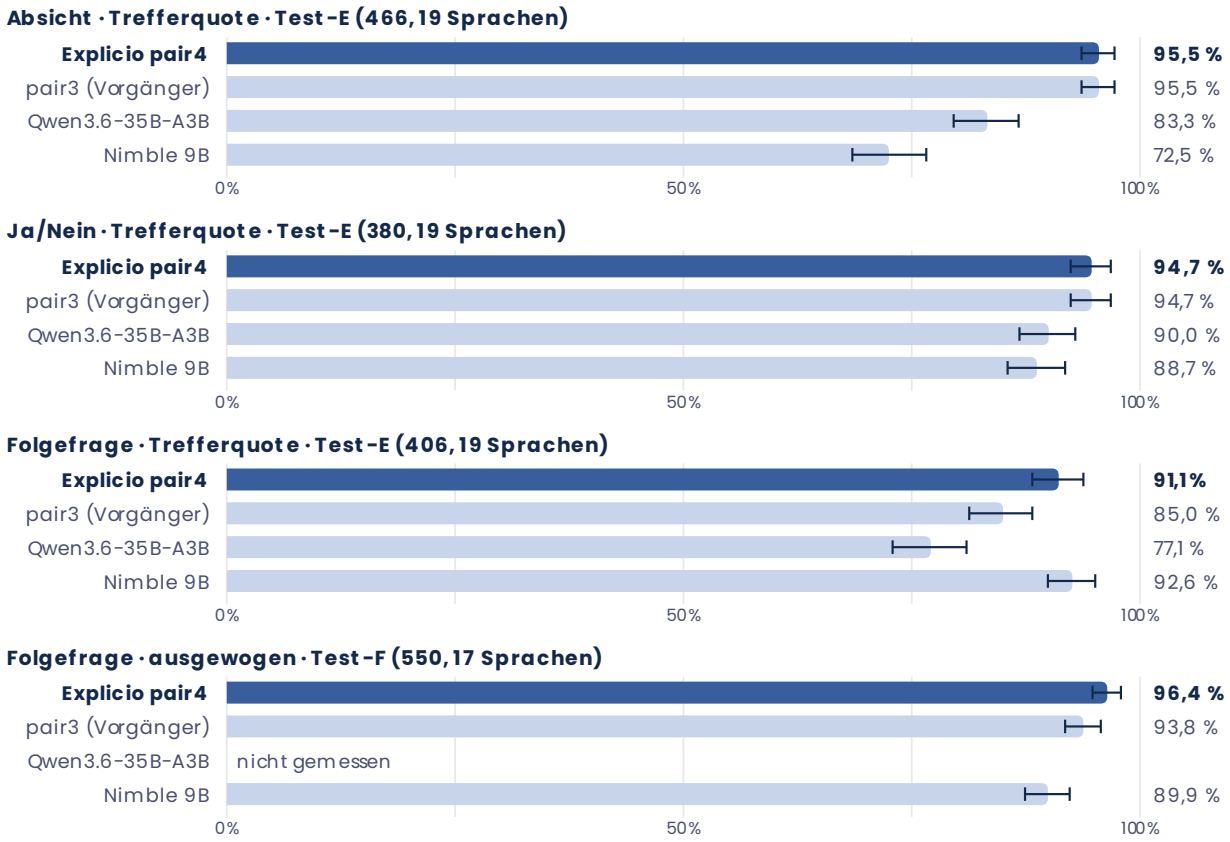
Die technische Pointe des Werkstattberichts ist eine Einsicht, die drei verworfene Versuche gekostet hat: Dem Encoder fehlte kein Wissen. Der Engpass war der Mittelwert über alle Wörter, mit dem der alte Ausgabekopf das Wissen abgelesen hat. Ein Aufmerksamkeits-Kopf, der die Wörter selbst liest, schließt den größten Teil der Lücke, ohne den Encoder anzufassen – und damit, ohne die beiden anderen Entscheidungen zu verändern.

Kennzahl	Wert	Herkunft
Absicht, Trefferquote, Test-E (466 Nachrichten, 19 Sprachen)	pair4 95,5 % · Qwen3.6 83,3 % · Nimble 72,5 %	`teste-pair4.ndjson`, E4-Rohdaten `qwen36-frei.ndjson`, `nimble-info.ndjson`

Kennzahl	Wert	Herkunft
Ja/Nein, Trefferquote, Test-E (380 Antworten, 19 Sprachen)	pair4 94,7 % · Qwen3.6 90,0 % · Nimble 88,7 %	dieselben Dateien
Folgefrage, Trefferquote, Test-E (406 Nachrichten, 19 Sprachen)	pair4 91,1 % · Nimble 92,6 % · pair3 85,0 % · Qwen3.6 77,1 %	dieselben Dateien, `s4b/pair3-teste.ndjson`
Folgefrage, ausgewogene Trefferquote, Test-F (550 Nachrichten, 17 Sprachen)	pair4 96,4 % · pair3 93,8 % · Nimble 89,9 %	`f2e/onnx/testf.ndjson`, `f2a/rohdaten/f2a-nimble-testf.ndjson`
Zeit je Entscheidung, Box-CPU, p95	62,8 ms (alle drei in einem Aufruf: 202,4 ms)	`f2e/box/bench-box-pair4.json`
Folgefrage: statistisch nachgewiesene Fehlerschranke bei 95 % Sicherheit	4,8 % bei 95,3 % Abdeckung	`f2e/gate/gate.json`
Absicht und Ja/Nein gegenüber pair3	bitgleich auf 3.700 Fällen (Δp = 0)	F2e-Bericht, Regel E2
Speicher des geladenen Modells	1.526 MiB	`f2e/box/bench-box-pair4.json`

Alle Zahlen stehen mit Quelle, Fallzahl, Messdatum und 95-%-Intervall im Zahlenblatt `docs/publikationen/pair4/zahlenblatt.json`, das ein Skript aus den Messzeilen zählt (`tools/publikationen/pair4_zahlen.py`). Alle Testsätze sind synthetisch.

ABB. 1 **Drei Entscheidungen auf den Freigabesätzen: Balken ab 0, Strich = 95-%-Intervall, alle Sätze synthetisch**



Balken ab 0, Strich = 95-%-Intervall (Bootstrap, 10.000 Ziehungen). Alle Sätze synthetisch.

KAPITEL 2

Warum vor jeder Suche drei Entscheidungen stehen



Explicio sammelt Wissen dort, wo es entsteht: in Chats, Sprachnachrichten, Fotos und Dokumenten. Es fragt nach, wenn etwas fehlt, und beantwortet Fragen mit Belegen aus einem Wissensgraphen. In diesem Ablauf fällt bei jeder Nachricht eine Reihe kleiner Entscheidungen, lange bevor ein Sprachmodell eine Antwort formuliert. Drei davon trifft heute pair4. Die Beispiele in diesem Abschnitt sind erfunden; Sätze aus den Testsätzen zitieren wir nicht.

Absicht (Intent). „Wer ist für die Kältemaschine in Halle 3 zuständig?“ ist eine Frage. „Die Kältemaschine in Halle 3 wartet ab sofort Team Nord.“ ist ein Wissensbeitrag. Dazu kommen Smalltalk, Fragen zum Einstieg, Antworten auf eine offene Rückfrage und Bitten, eine Person zu bewerten – die Explicio erkennt und ablehnt. Liegt die Entscheidung daneben, landet eine Frage als Wissen im Graphen oder ein Beitrag wird als Frage beantwortet.

Ja/Nein. Explicio fragt zurück, etwa: „Gehören diese beiden Einträge zusammen?“ Die Antwort kommt selten als „ja“. Sie lautet „passt schon“, „eher nicht, das ist die alte Anlage“ oder „keine Ahnung“. Wie sie gemeint ist, entscheidet, ob zwei Wissensstücke verknüpft werden. Eine Wortliste, die nach „ja“ und „nein“ sucht, trifft auf unserem Testsatz 34,5 %.

Folgefrage. Nach „Wie hoch ist der Prüfdruck der Pumpe P-12?“ kommt „Und bei der P-14?“. Diese Nachricht ist allein nicht durchsuchbar; sie braucht den Verlauf und wird vor der Suche umgeschrieben. „Wie lange dauert die Einweisung für neue Kolleginnen?“ dagegen steht für sich, und ein Umschreiben würde sie mit Pumpen vermischen. Ein übersehenes Umschreiben kostet die Antwort; ein unnötiges kostet einen Aufruf und im schlechten Fall die Treffergenauigkeit. Die Regel, die bisher im Betrieb entscheidet – Listen mit Pronomen und Verweiswörtern, dazu Länge und Anfangswort –, erreicht auf unseren Kalibriersätzen eine ausgewogene Trefferquote von 57,6 %.



KURZ GESAGT

Die Qualität einer Antwort entscheidet sich zum Teil, bevor gesucht wird. Drei kleine Entscheidungen bestimmen, womit gesucht wird und was in den Wissensgraphen gelangt. Sie müssen schnell, mehrsprachig und überprüfbar sein.

Vier Wege zu einer Entscheidung

Für diese Art von Entscheidung gibt es heute vier verbreitete Ansätze. Wir haben alle vier gebaut oder gemessen. Die folgende Übersicht ist ein Leitfaden; die Zahlen stammen aus unseren Messungen auf unseren Testsätzen und gelten für diese.

Regeln und Wortlisten

Eine Liste von Wörtern und Mustern, je Sprache gepflegt. So hat Explicio angefangen, und für die eindeutigen Fälle bleibt eine Wortliste vorgeschaltet.

Vorteile	Nachteile
sofort, ohne Rechenaufwand	trifft auf unseren Sätzen 34,5 % (Ja/Nein) bzw. 57,6 % ausgewogen (Folgefrage)
vollständig nachvollziehbar	braucht Listen je Sprache; Sprachen, die Pronomen weglassen, entziehen sich
keine Trainingsdaten	kennt keinen Kontext: „und …“ gilt pauschal als Folgefrage

Großes Sprachmodell mit Anweisung

Ein großes Sprachmodell bekommt die Nachricht und eine Beschreibung der Klassen und wählt eine davon. Gemessen haben wir Qwen3.6-35B-A3B, das Modell, das auf unseren Boxen die Antworten schreibt, als Klassifikator mit genau einem Ausgabe-Token: je Klasse ein Buchstabe, gewählt wird der wahrscheinlichste.

Vorteile	Nachteile
keine Trainingsdaten; eine neue Klasse ist eine neue Zeile in der Beschreibung	ohne Training in allen drei Entscheidungen unter unserem kleinen Modell: 83,3 % / 90,0 % / 77,1 % auf Test-E
läuft auf vorhandener Hardware, kein neuer Dienst	braucht die GPU, die auch die Antworten schreibt; 151 ms je Entscheidung (p95, über das Netz)
versteht viele Sprachen	schlechter kalibriert: seine Sicherheit sagt wenig darüber, wie oft es recht hat (Kalibrierfehler bis 0,144 gegen höchstens 0,048 bei pair4)

Spezialisiertes Sprachmodell eines Drittanbieters

Inzwischen bieten Anbieter kleine Sprachmodelle an, die für genau solche Entscheidungen feinabgestimmt sind. Gemessen haben wir Nimble 9B von Bespoke Labs (Apache-2.0) und wollten Clef Flash von Cloudflare messen.

Vorteile	Nachteile
ohne eigenes Training stark bei der Folgefrage: 92,6 % auf Test-E	belegt unter Ollama rund 16 GB und ist auf der Box neben dem Antwortmodell nicht betreibbar; Clef Flash scheiterte schon beim Laden
offene Lizenz (Apache-2.0), veröffentlichte Trainingsdaten	524 ms je Entscheidung (p95, GPU, ohne Netzweg)

Vorteile	Nachteile
	bei der Absicht schwach (72,5 %), auf dem neuen Satz Test-F hinter unserem Vorgänger

Kleiner Encoder, selbst trainiert

Ein Encoder liest Nachricht und Kontext gemeinsam und gibt für jede Entscheidung Wahrscheinlichkeiten aus. Das ist der Weg von pair3 und pair4.

Vorteile	Nachteile
auf der CPU: 62,8 ms je Entscheidung, 1,5 GiB Speicher, die GPU bleibt frei	braucht Trainingsdaten; jede neue Aufgabe ist ein neuer Trainingslauf
kalibriert: die Sicherheit trägt eine statistisch nachgewiesene Fehlergrenze	die Daten sind synthetisch; wie gut sie echten Verkehr treffen, zeigt erst der Betrieb
ein Modell, ein Pin, deterministisch auf der CPU	kennt genau seine Aufgaben, nichts darüber hinaus



KURZ GESAGT

Für drei feste, häufige Entscheidungen ist ein kleiner, eigens trainierter Encoder auf unseren Sätzen genauer, schneller und besser abgesichert als ein großes Sprachmodell. Das große Modell bleibt im Vorteil, wo sich die Aufgabe ständig ändert.

KAPITEL 3

Wie pair4 entstand



Ausgangspunkt: pair3

pair3 ist ein Satzpaar-Encoder auf `jhu-clsp/mmBERT-base`, einem mehrsprachigen Encoder unter MIT-Lizenz: 22 Schichten, 307 Millionen Parameter, davon 197 Millionen in der Einbettung des Vokabulars von 256.000 Token. Die Eingabe ist eine Sequenz aus Kontext und Nachricht – `[CLS] Kontext [SEP] Nachricht [SEP]`, höchstens 320 Token –, sodass jede Schicht Aufmerksamkeit über beide Teile rechnet. Auf dem letzten Zustand sitzt für jede Entscheidung ein linearer Kopf, gespeist aus dem Mittelwert über alle Token.

Trainiert wurde pair3 dreimal unabhängig mit verschiedenen Startwerten auf der GPU der Boxen; ausgeliefert wird das arithmetische Mittel der Gewichte. Das Mitteln bringt messbar etwas: Die Einzelläufe treffen die Absicht auf den Kalibriersätzen zu 93,0 bis 93,5 %, das Mittel zu 94,2 %. Jede Entscheidung bekommt eine Temperatur, und für Ja/Nein und Folgefrage gibt es ein Gate: eine Schwelle, oberhalb derer das Modell selbst entscheidet, gewählt so, dass die Fehlerquote mit 95 % Sicherheit unter einer festen Grenze liegt.

Die Trainingsdaten sind zweierlei. Der erste Teil wurde mit Claude geschrieben, der zweite vom Antwortmodell unserer eigenen Boxen über das Explicio Grid erzeugt, die Klasse aus einer Platzliste vorgegeben und von Claude-Subagenten blind geprüft. Zusammen sind es 2.423 Beispiele für die Absicht, 3.137 für Ja/Nein und 1.820 für die Folgefrage, in 18 Sprachen, alle gegen sämtliche Testsätze bereinigt.

pair3 hatte eine klare Schwäche. Nachrichten, die zum selben Thema gehören, aber für sich verständlich sind – ein Themenwechsel mit „und“, ein Pronomen, das sich in derselben Nachricht auflöst, die Höflichkeitsform, Stichworte –, hielt es oft für abhängig. Auf Test-E erkannte es nur 64,9 % der eigenständigen Nachrichten. Die Reihenfolge stimmte (ROC-AUC 0,979), die Grenze lag zu weit auf der Seite „abhängig“.

Methode

Ein Lastenheft vor dem Bau

Bevor pair4 gebaut wurde, stand fest, was es leisten muss: ein Lastenheft mit Muss- und Soll-Werten für jede Entscheidung, für Sprachen, Kalibrierung, Tempo, Speicher, Lizenz und Daten. Die wichtigsten Muss-Werte: Absicht und Ja/Nein nicht schlechter als pair3; die Folgefrage auf dem neuen Freigabesatz Test-F mit mindestens 30 % weniger Fehlern bei der ausgewogenen Trefferquote, also mindestens 95,625 %; Soll waren 40 % weniger Fehler, 96,25 %. Die abhängigen Nachrichten mindestens zu 98 %. Und der Betriebsrahmen von pair3: ein Modell, ein Pin, CPU, höchstens 1,6 GiB.

Zwei Grenzen haben wir im Lauf verschoben, und beide stehen hier. Die Schwelle für Test-F war ursprünglich 40 % weniger Fehler; sie wurde vor der Messung auf 30 % gesenkt, weil schon die Hochrechnung für ein vollständig trainiertes Modell im Rauschen der alten Schwelle lag. Die Zeitgrenze auf der Box lag bei 200 ms für einen Aufruf mit drei Entscheidungen; pair4 kam auf 202,4 ms, und die Grenze wurde **nach** der Messung auf 210 ms gesetzt. Die betriebliche Vorgabe für diesen Rechenort war 300 ms; die 200 ms waren als Rahmen von pair3 mit 3 % Abstand entstanden.

Regeln, die vor der Messung feststehen

Kriterien zuerst. Für jede Session liegen Schwellen und Auswerteskript in der Versionsverwaltung, bevor die erste Zahl entsteht. Eine Regel darf nach dem Ergebnis geändert werden, wenn der Fehler nachweislich in der Regel liegt; dann als Nachtrag mit Begründung, und kein Tor gilt dadurch als bestanden.

Freigabesätze einmal. Test-E und Test-F sieht jedes Modell genau einmal, auf ausdrücklichen Namen. Sie liegen nie im Training, nie in der Kalibrierung, nie im Gate, und niemand sieht sich einzelne Sätze an, um nachzubessern. Gewählt wird auf den Kalibriersätzen B, C und D.

Synthetisch und blind geprüft. Test-E haben Claude-Subagenten (Claude Sonnet) nach Regeln geschrieben, blind geprüft von Claude Opus. Test-F haben zwölf Instanzen von Claude Opus geschrieben, je eine Branche und höchstens drei Sprachen; sechs Instanzen von Claude Sonnet haben blind geprüft und stimmten in 574 von 576 Fällen überein. Eine Autoreninstanz, die die Datei einer anderen geöffnet hatte, wurde verworfen und ersetzt.

Gepaart und mit Intervall. Unterschiede zwischen zwei Modellen rechnen wir gepaart, Beispiel für Beispiel, mit einem 95-%-Intervall über 10.000 Bootstrap-Ziehungen; bei der ausgewogenen Trefferquote je Klasse geschichtet.

Gemessen wie im Betrieb. pair4 wird als derselbe ONNX-Graph gemessen, der in der Cloud und auf der Box läuft. Die Zeit stammt von einer Box, im selben Lauf wie pair3.

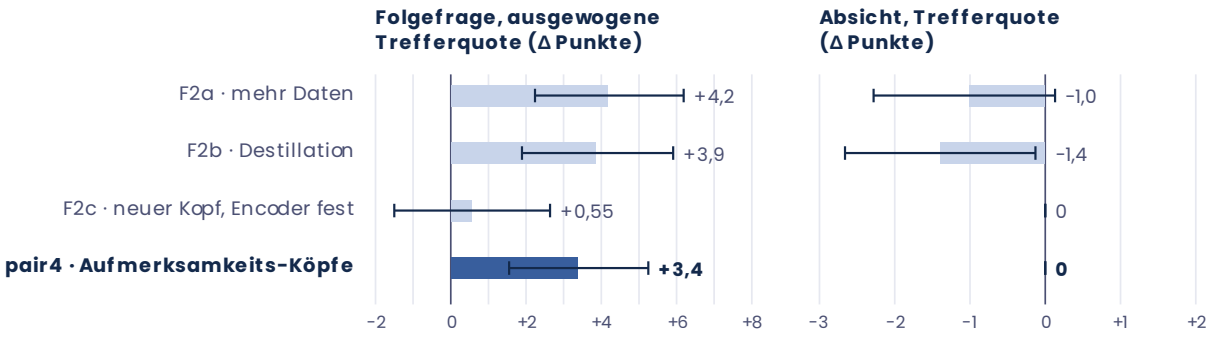
Mindestens drei Läufe. Jeder Arm wird mindestens dreimal mit verschiedenen Startwerten trainiert, ausgeliefert wird das Mittel, und jede Änderung läuft gegen eine Kontrolle gleicher Menge und Schrittzahl.

Der Weg zu pair4

Vom ersten Versuch am 24. September bis zur Abnahme von pair4 am 5. Oktober:

Datum	Schritt
24./25. September	Erste Versuche auf einem fertigen mehrsprachigen Klassifikator; in seinem Stammbaum steht ein nicht-kommerzieller Datensatz. Neustart auf mmBERT-base (MIT)
26. September	pair1, das erste eigene Satzpaar-Modell, im Schattenbetrieb
27. September	pair2: Mittel dreier Läufe statt eines einzelnen
28. September	pair3 besteht den Freigabesatz Test-E; die Lesart von Ja/Nein-Antworten wirkt seitdem als Rückfrage an den Menschen
1. Oktober	pair3 läuft auf der CPU der Boxen; ein kleineres Basismodell wird gemessen und verworfen (Absicht -8,2 Punkte)
4./5. Oktober	fertige Entscheidungsmodelle gemessen: Qwen3.6 und Nimble; Clef Flash nicht messbar
5. Oktober	F2a bis F2c verworfen, F2d findet den Engpass, F2e baut pair4; abgenommen und in der Cloud und auf der ersten Box eingesetzt

ABB. 2 **Gewinn bei der Folgefrage, Verlust bei der Absicht**



Gepaart gegen pair3 auf den Kalibriersätzen B+C+D (589 bzw. 790 Fälle), Strich = 95-%-Intervall.

Alle Werte gepaart gegen pair3 auf den Kalibriersätzen B+C+D, Folgefrage als ausgewogene Trefferquote, Absicht als Trefferquote. Beleg: `weg_bcd` im Zahlenblatt.

F2a: mehr Daten, mit einem Lehrer

Die Schwäche bei eigenständigen Nachrichten sah nach einem Datenloch aus. Wir haben 1.667 neue Kandidaten für die Folgefrage vom Antwortmodell unserer Boxen schreiben lassen und Nimble als Lehrer darüber gesetzt: Es bestätigte 98 % der abhängigen, aber nur 38 % der eigenständigen Kandidaten – und behielt in der Stichprobe meist recht. Übrig blieben 963 Beispiele. Mit ihnen weitertrainiert stieg die Folgefrage um 4,2 Punkte. Die Absicht fiel um 1,0 Punkte. Ein Encoder für drei Aufgaben: Was der einen hilft, verschiebt die anderen. Nach den vorab festgelegten Regeln („nichts anderes wird schlechter“) kein Wechsel.

F2b: den Vorgänger festhalten

Wenn sich der Encoder ändert, könnten Absicht und Ja/Nein vom Vorgänger lernen, wie sie zu antworten haben – Destillation mit pair3 als Lehrer. Die Folgefrage stieg um 3,9 Punkte, die Absicht fiel trotzdem um 1,4. Der Grund steht in den Zahlen des Lehrers: Auf seinen eigenen Trainingsdaten ist pair3 fast immer sicher (Median 0,996). Weiche Ziele, aus denen ein Schüler etwas lernen könnte, gibt es dort kaum. Kein Wechsel.

F2c: den Encoder einfrieren

Wenn sich der Encoder nicht ändern darf, bleibt nur der Kopf. Wir haben den linearen Kopf der Folgefrage auf dem eingefrorenen pair3 neu gelernt, mit allen Daten. Absicht und Ja/Nein blieben bitgleich, auf 3.700 Fällen dieselben Wahrscheinlichkeiten. Die Folgefrage gewann aber nur +0,55 Punkte, und die ROC-AUC sank: Der neue Kopf verschob die Grenze, trennte die schweren Fälle aber nicht besser. Kein Wechsel.

F2d: die Einsicht

Drei Versuche, ein Muster: Wo der Encoder mitlernt, gewinnt die Folgefrage und die Absicht verliert. Wo er fest bleibt, verliert niemand, aber es gewinnt auch niemand. Die naheliegende Deutung wäre, dass die 110 Millionen Encoder-Parameter jenseits der Einbettung für drei Aufgaben nicht reichen.

Die Messungen sprechen dagegen. Mit lernendem Encoder erreicht derselbe lineare Kopf eine ausgewogene Trefferquote von 94,5 % – die Bezüge zwischen Verlauf und Nachricht lassen sich also darstellen. Und pair3 ist ein Cross-Encoder: Jede seiner 22 Schichten rechnet Aufmerksamkeit über Verlauf und Nachricht zugleich. Was F2c zeigt, ist enger: Der **eingefrorene** Encoder mit **Mittelwert** über alle Token und linearem Kopf reicht nicht. Der Mittelwert ist der Verdächtige. Er fasst zwanzig oder zweihundert Token-Zustände zu einem Vektor zusammen und verwischt dabei, welches Wort sich worauf bezieht. Ob „sie“ in der neuen Nachricht auf etwas im Verlauf zeigt oder auf etwas in derselben Nachricht, steht in den Zuständen der einzelnen Token – nach dem Mitteln nicht mehr.

Also haben wir den Mittelwert ersetzt, nicht den Encoder. Der neue Kopf ist ein kleiner Aufmerksamkeits-Kopf mit 0,89 Millionen Parametern: Die Token der neuen Nachricht fragen die Token des Verlaufs ab, über die unveränderten Zustände des eingefrorenen pair3. Ergebnis auf den Kalibriersätzen: ausgewogene Trefferquote 93,96 % statt 90,61 %, ROC-AUC 0,985 statt 0,979. Der Kopf schließt 83 % der Lücke zwischen dem linearen Kopf aus F2c und dem vollständig trainierten Encoder aus F2a – ohne einen einzigen Parameter des Encoders zu ändern.



KURZ GESAGT

Dem Encoder fehlte kein Wissen. Der Engpass war die Art, es abzulesen. Ein Kopf, der die Wörter selbst liest, holt aus derselben Darstellung, was der Mittelwert verwischt – und lässt alles andere unberührt.

F2e: pair4

Aus dem Vortest wurde ein Modell. Drei solcher Köpfe, mit verschiedenen Startwerten gelernt, sitzen im ONNX-Graphen von pair4; ihre Wahrscheinlichkeiten werden gemittelt und mit einer gemeinsamen Temperatur von 0,88 kalibriert. Absicht und Ja/Nein laufen unverändert über die Köpfe von pair3.

Die Abnahme folgte dem Lastenheft. Export-Parität: 774 von 774 Fällen mit gleicher Wahl, Abweichung höchstens $3 \cdot 10^{-6}$. Absicht und Ja/Nein bitgleich zu pair3 auf 3.700 Fällen. Folgefrage auf Test-F: ausgewogen 96,42 % – das Soll von 96,25 % ist erfüllt. Abhängige Nachrichten 99,63 %, ein Fehler unter 270. Kalibrierfehler (ECE) 0,019. Die einzige verfehlte Zeile war die Zeitgrenze auf der Box (202,4 statt 200 ms); sie ist im Abschnitt „Methode“ beschrieben.

Architektur

ABB. 3 **Vom Mittelwert zum Aufmerksamkeits-Kopf**

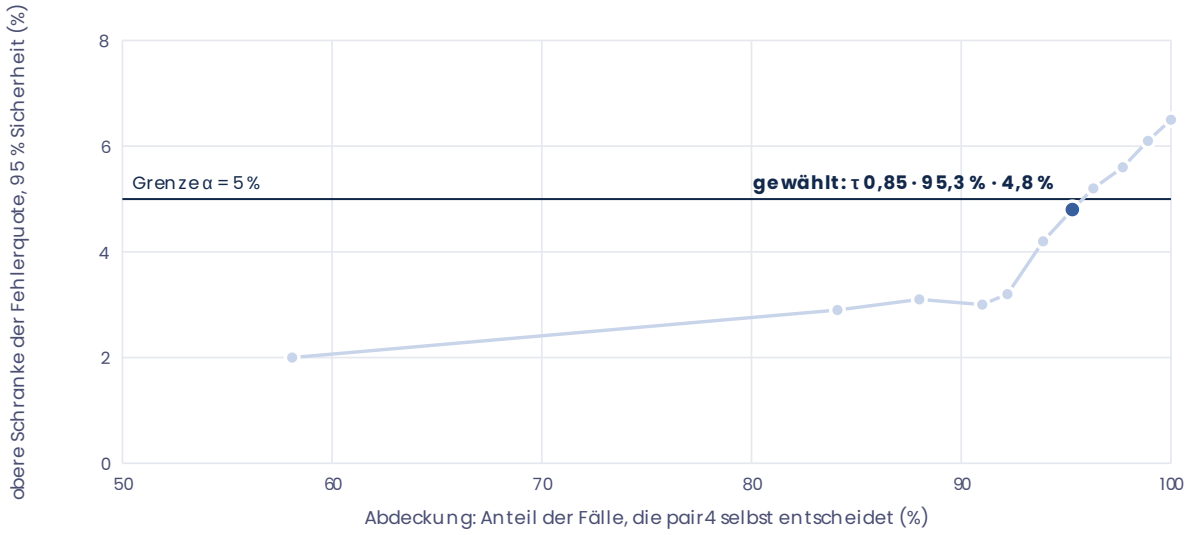


Ein ONNX-Graph, gleiche Ein- und Ausgänge wie pair3. Der Mittelwert verwischt, welches Wort sich worauf bezieht; der Kopf liest die Token selbst.

pair4 ist ein einziger ONNX-Graph mit denselben Ein- und Ausgängen wie pair3. Die Laufzeit im `ai-gateway` musste nicht wissen, dass sich etwas geändert hat: Sie liest für Folgefrage-Zeilen einen anderen Ausgang. Die Masken, die Verlauf und Nachricht im Kopf trennen, entstehen im Graphen aus den Positionen der Trennzeichen. Der Kopf rechnet für jede Zeile eines Aufrufs mit, auch für Zeilen der Absicht und von Ja/Nein, deren Ergebnis die Laufzeit verwirft. Das kostet rund 3 % Zeit und ist der Grund für die 202,4 ms.

Das Gate der Folgefrage ist neu berechnet. Bei einer Schwelle von 0,85 entscheidet pair4 95,3 % der Fälle selbst; die beobachtete Fehlerquote liegt dort bei 3,0 %, die obere Schranke mit 95 % Sicherheit bei 4,8 %, unter der vorgegebenen Grenze von 5 %. Bei pair3 lag die Schwelle bei 0,93 und deckte 81,3 % ab. Grundlage sind die Kalibrier-Scheibe und die Sätze A bis D, synthetisch; das Gate ist deshalb vorläufig, bis etikettierter Verkehr die endgültige Schwelle liefert.

ABB. 4 **Abdeckung gegen statistisch nachgewiesene Fehlerschranke**



Je Punkt eine Schwelle τ (0,50 bis 0,99; höhere Schwellen decken unter 5 % ab). Grundlage: Kalibrier-Scheibe und Test-A–D, synthetisch, vorläufig.

Je Punkt eine Schwelle τ. Beleg: `gate_pair4.rewrite.kurve` im Zahlenblatt, aus `f2e/gate/gate.json`.

KAPITEL 4

Ergebnisse



Die drei Entscheidungen im Vergleich

Abbildung 1 in der Zusammenfassung zeigt die Werte als Balken; die Tabelle nennt beide Maße.

Entscheidung, Satz, Maß	pair4	pair3	Qwen3.6-35B-A3B	Nimble 9B
Absicht, Test-E, Trefferquote	**95,5** [93,6; 97,2]	95,5	83,3 [79,6; 86,7]	72,5 [68,5; 76,6]
Absicht, Test-E, ausgewogen	**95,4**	95,4	85,8	83,4
Ja/Nein, Test-E, Trefferquote	**94,7** [92,4; 96,8]	94,7	90,0 [86,8; 92,9]	88,7 [85,5; 91,8]
Ja/Nein, Test-E, ausgewogen	**94,7**	94,7	90,0	88,6
Folgefrage, Test-E, Trefferquote	91,1 [88,2; 93,8]	85,0	77,1 [72,9; 81,0]	**92,6** [89,9; 95,1]
Folgefrage, Test-E, ausgewogen	89,7	82,5	73,3	**91,5**
Folgefrage, Test-F, ausgewogen	**96,4** [94,8; 97,9]	93,8 [91,8; 95,7]	nicht gemessen	89,9 [87,4; 92,3]
Folgefrage, Test-F, Trefferquote	**96,4** [94,7; 97,8]	93,6 [91,5; 95,6]	nicht gemessen	89,8 [87,3; 92,2]

Werte in Prozent, in Klammern das 95-%-Intervall. Beleg: `zellen` im Zahlenblatt. Mit dem jeweils anderen Maß ändert sich die Reihenfolge in keiner Zeile.

Gepaart gerechnet liegt pair4 bei der Absicht 12,2 Punkte vor Qwen3.6 (Intervall 8,6 bis 15,9) und 23,0 Punkte vor Nimble (18,7 bis 27,3), bei Ja/Nein 4,7 Punkte vor Qwen3.6 (1,6 bis 7,9) und 6,1 vor Nimble (2,9 bis 9,5). Bei der Folgefrage auf Test-F liegt pair4 6,5 Punkte vor Nimble (4,0 bis 9,1) und 2,7 Punkte vor pair3 (1,4 bis 4,1, ausgewogen).

Bei Nimbles Absicht ist die Lücke zwischen Trefferquote (72,5 %) und ausgewogener Trefferquote (83,4 %) auffällig: Es verfehlt vor allem die große Klasse der Bitten um eine Personenbewertung, die es oft für Antworten auf eine Rückfrage hält; kleinere Klassen trifft es deutlich besser. Die Reihenfolge der Modelle ändert das nicht.

Die Unterschiede liegen dort, wo pair3 schwach war. Auf Test-F erkennt pair4 93,2 % der eigenständigen Nachrichten, pair3 87,5 %, Nimble 82,9 %. Bei den abhängigen Nachrichten verfehlt pair4 eine von 270, pair3 keine, Nimble acht.

Wo andere besser sind

Nimble bei der Folgefrage auf Test-E. Nimble trifft dort 92,6 %, pair4 91,1 %. Gepaart ist der Abstand -1,5 Punkte mit einem Intervall von -4,9 bis +2,0: Er liegt im Rauschen, spricht aber nicht für uns. Auf dem neueren Satz Test-F kehrt sich das Bild um. Dass Nimble auf Test-E besser und auf Test-F schlechter abschneidet, deutet auf eine Stilabhängigkeit der Testsätze (Test-E von Claude Sonnet, Test-F von Claude Opus geschrieben). Ein einzelner Satz trägt eine Aussage über „besser“ nicht.

Neue Aufgaben. Ein großes Sprachmodell lernt eine neue Kategorie aus einer Beschreibung. pair4 kennt genau seine drei Aufgaben; eine vierte braucht Daten und einen Trainingslauf.

Die GPU. Auf der GPU der Box rechnete ein Modell dieser Bauweise alle drei Entscheidungen in 10 ms (p95), rund zwanzigmal schneller als auf der CPU. Wir rechnen auf der CPU, weil die GPU die Antworten schreibt und ein Speichertopf beide versorgt.

Der Vorgänger. pair3 erkannte auf Test-F alle 270 abhängigen Nachrichten und war dort etwas besser kalibriert (ECE 0,012 gegen 0,019).

Tempo und Rechenort

Modell	Rechenort	p50	p95	Fälle
pair4, eine Entscheidung je Aufruf	CPU der Box, 2 Fäden	39,3 ms	62,8 ms	665
pair4, drei Entscheidungen in einem Aufruf	CPU der Box, 2 Fäden	131,5 ms	202,4 ms	221
pair3, eine Entscheidung je Aufruf	CPU der Box, 2 Fäden	38,1 ms	60,9 ms	665
Qwen3.6-35B-A3B, eine Entscheidung je Aufruf	GPU der Boxen, Rundreise über das Netz	133,0 ms	151,0 ms	1.252
Nimble 9B, eine Entscheidung je Aufruf	GPU der Box, ohne Netzweg, ohne Ladezeiten	292,0 ms	523,5 ms	1.042

Beleg: `tempo_box_cpu` und `tempo_sprachmodelle` im Zahlenblatt.

Die Zeiten der Sprachmodelle stammen aus denselben Messläufen wie ihre Trefferquoten, auf denselben drei Aufgaben von Test-E. Gleiche Bedingungen sind es nicht: Die Sprachmodelle rechnen auf der GPU, Qwen3.6 mit dem Weg über das Netz, Nimble ohne; pair4 rechnet auf der CPU, und seine Zeitmessung nutzt 225 Referenzfälle statt der Freigabesätze. Die Tabelle zeigt, was jedes Modell dort kostet, wo es laufen würde. pair4 belegt 1.526 MiB Speicher und keinen Grafikspeicher.

KAPITEL 5

Betrieb und Vertrauen



Betrieb

pair4 läuft im `ai-gateway` auf der CPU, in der Cloud und auf der Box, mit ONNX Runtime und zwei Fäden. Auf der Box kommt es als eigenes Image mit dem signierten Release, schreibgeschützt eingehängt, und gelangt über die Update-Ringe auf jede Box. In der Cloud und auf der ersten Box läuft es seit dem 5. Oktober 2026.

Wirkung im Betrieb hat bisher nur Ja/Nein, und zwar als Rückfrage: Erkennt die Wortliste nichts und ist das Modell über seiner Schwelle sicher, fragt Explicio mit seiner Lesart noch einmal nach, und gewirkt wird erst mit der ausdrücklichen Antwort des Menschen. Eine Wirkung ohne Rückfrage ist je Mandant erst nach einem eigenen statistischen Tor möglich; umgeschaltet ist bisher kein Mandant. Absicht und Folgefrage laufen im Schatten mit: Ihre Entscheidungen werden aufgezeichnet und mit dem heutigen Weg verglichen, aber nicht angewandt. Ob und wann die Folgefrage wirkt, ist ein eigener Entscheid mit eigenem Tor.

Weil Absicht und Ja/Nein bitgleich zu pair3 sind, hat der Wechsel nichts verändert, was heute wirkt. Der Rückweg ist ein Zeiger und ein Neustart; der Ordner von pair3 bleibt liegen.

Vertrauen

Keine Kundendaten. pair4 ist ausschließlich mit Daten trainiert, die wir selbst erzeugt haben: mit Claude geschrieben, vom Antwortmodell unserer eigenen Boxen erzeugt, von Nimble gefiltert, blind geprüft und gegen alle Testsätze bereinigt. Kein Kundenwissen ist in ein Modell geflossen. Explicio lernt nicht über Mandanten hinweg; die Daten eines Kunden verbessern nie ein Modell, das ein anderer Kunde nutzt. Ob mit Claude erzeugte Daten für das Training eines Entscheiders innerhalb unseres Produkts verwendet werden dürfen, haben wir vorab geklärt: Ein solcher Entscheider ist kein Konkurrenzprodukt.

Lizenzen. Das Basismodell mmBERT-base steht unter MIT-Lizenz; in seinem Stammbaum ist kein nicht-kommerzieller Datensatz genannt. Einen ersten Kandidaten haben wir am 25. September verworfen, weil seine Trainingsdaten einen nicht-kommerziellen Datensatz enthielten. Jedes Vergleichsmodell haben wir vor der Messung geprüft; Nimble und Clef stehen unter Apache-2.0, ein weiteres Modell ohne angegebene Lizenz der Gewichte trat nicht an. Die ausgelieferten Lizenztexte der Box-Images sind Gegenstand eines laufenden Lizenzpakets.

Einstufung nach der KI-Verordnung. pair4 ordnet Nachrichten ein. Es bewertet keine Person, wählt niemanden aus und entscheidet nicht über Menschen; Bitten, eine Person zu bewerten, erkennt es als solche, und Explicio lehnt sie ab. Explicio bleibt ein Werkzeug der Wissensverwaltung; die Einstufung ändert sich durch pair4 nicht, und jede Neubewertung steht im KI-Systemregister des Unternehmens. Jede Handlung mit Wirkung bleibt bei einem Menschen.

Nachvollziehbarkeit. Jedes ausgelieferte Modell trägt ein Manifest mit Prüfsummen und ist über einen Pin im Code festgelegt. Startwerte, Trainingspläne, Kriterien-Dateien mit Commit und Messzeilen liegen im Repository; jeder Lauf lässt sich mit denselben Daten, Startwerten und Skripten wiederholen.

KAPITEL 6

Grenzen und Ausblick



Grenzen

- **Synthetische Sätze.** Alle Trainings- und Testsätze sind von Modellen geschrieben. Die Prüfer waren sich fast immer einig (574 von 576 bei Test-F), die Sätze enthalten also eher eindeutige Fälle; echter Verkehr ist unschärfer. Wie gut pair4 dort trifft, zeigt erst der Betrieb.
- **Ein Satz trägt keinen Vergleich allein.** Nimble ist auf Test-E besser, auf Test-F schlechter als pair4. Die Aussage „pair4 ist besser als Nimble“ gilt für die Absicht, für Ja/Nein und für Test-F, nicht für die Folgefrage auf Test-E.
- **Das Sprachmodell als Ein-Token-Klassifikator.** Gemessen ist Qwen3.6 mit einem Ausgabe-Token je Entscheidung, nicht mit der ausführlichen Anweisung, die Explicio im Produkt für andere Aufgaben nutzt. Eine ausführlichere Anweisung könnte besser abschneiden und wäre langsamer.
- **Clef Flash nicht gemessen.** Unter Ollama fiel der freie Speicher der Box schon beim Laden unter unsere Sicherheitsschwelle. Messbar wäre es auf mehr Speicher oder über eine andere Laufzeit.
- **Zeit unter verschiedenen Bedingungen.** Die Zeiten der Sprachmodelle und von pair4 stammen aus verschiedenen Rechenwegen und Fallmengen; die Tabelle vergleicht, was jedes Modell an seinem Ort kostet, kein Wettrennen unter gleichen Bedingungen.
- **Eine Box, eine Messung.** Die Zeit von pair4 ist auf einer Box zu einem Zeitpunkt gemessen. Die Zeitgrenze wurde nach der Messung verschoben.
- **Vorläufiges Gate.** Die statistisch nachgewiesene Schranke gilt für die synthetischen Kalibriersätze. Die endgültige Schwelle kommt aus etikettiertem Verkehr.
- **Schatten.** Die Folgefrage wirkt noch nicht. Die Zahlen beschreiben, was pair4 entscheiden würde.

Was als Nächstes kommt: Explicio pair5

pair5 ist in Entwicklung, und seine Ziele stehen fest, bevor es gebaut ist: 95 % je Entscheidung auf einem neuen, unabhängigen Freigabesatz, mindestens 93 % in jeder großen Sprache mit höchstens 5 Punkten Gefälle, und alle Entscheidungen einer Nachricht in 40 ms auf der CPU der Box. Die Wege dahin sind im Forschungsprogramm beschrieben: Adapter, die nur für eine Entscheidung wirken; eine Kaskade, die eindeutige Fälle in Mikrosekunden entscheidet und nur die schweren an den Encoder gibt; und langfristig ein eigenes Basismodell, vortrainiert auf Text eines frei lizenzierten Modells. Gemessen wird nach denselben Regeln: Kriterien zuerst, Freigabesätze einmal, und der Vergleich sagt, wo andere besser sind.

Fünf Fragen an jeden Anbieter

Wer einen Wissensassistenten auswählt, kann diese Fragen jedem Anbieter stellen, auch uns.

1. **Welche Entscheidungen fallen vor der Suche, und wer trifft sie?**
   Eine Regel, ein großes Sprachmodell, ein eigenes Modell?

2. **Auf welchen Sätzen ist gemessen, und wie oft hat das Modell sie gesehen?**
   Eine Zahl auf einem Satz, der beim Entwickeln mitgelesen wurde, sagt wenig.

3. **Gibt es eine statistisch nachgewiesene Fehlergrenze, und was passiert darunter?**
   Ein Modell, das seine Unsicherheit nicht kennt, kann sie nicht abgeben.

4. **Wo rechnet die Entscheidung, und was kostet sie?**
   Auf welcher Hardware, mit welcher Zeit, neben welchen anderen Aufgaben?

5. **Wo ist ein anderes Modell besser?**
   Ein Vergleich, der das nicht sagt, ist eine Anzeige.

KAPITEL 7

Herkunft und Reproduzierbarkeit



Bestandteil	Fassung
Modell	`explicio-dm-pair4`, Manifest-sha256 `dbf3b079…`, Pin `services/ai-gateway/app/decide/pin.json`
Basismodell	`jhu-clsp/mmBERT-base`, Revision `c5955035…`, MIT
Köpfe der Folgefrage	drei Köpfe, Seeds 20261024–26, `koepfe.json` mit Prüfsummen
Laufzeit	ONNX Runtime 1.28 auf der CPU, 2 Fäden
Vergleichsmodelle	`qwen3.6-35b-a3b-nvfp4` (Qwen3.6-35B-A3B in NVFP4-Quantisierung) über vLLM; Nimble 9B (Ollama-Tag `nimble:9b`, LoRA auf Qwen3.5-9B) über Ollama 0.35.1; gemessen am 4. und 5. Oktober 2026
Lastenheft	`docs/lastenheft-entscheidungsmodell.md`
Kriterien und Berichte	`docs/messungen/entscheidungsmodelle/2026-10-04-e4-*`, `2026-10-05-f2a-*` bis `2026-10-05-f2e-*`, je `-kriterien.md` mit Commit vor der Messung
Messzeilen	`f2e/onnx/`, `f2e/box/`, `f2e/gate/`, `f2a/rohdaten/`, `2026-10-04-e4-rohdaten/`, `s4b/` unter `docs/messungen/entscheidungsmodelle/`
Zahlenblatt	`docs/publikationen/pair4/zahlenblatt.json`, erzeugt von `tools/publikationen/pair4_zahlen.py`
Abbildungen	`docs/publikationen/pair4/figures/`, erzeugt von `tools/publikationen/pair4_figures.py`

Die Freigabesätze selbst geben wir nicht heraus, damit sie für künftige Modelle unverbraucht bleiben. Messzeilen mit Kennung, Klasse, Vorhersage und Wahrscheinlichkeit, das Zahlenblatt und die Werkzeuge stellen wir Auditoren, Kunden und Hochschulen auf Anfrage zur Verfügung.

Über dieses Dokument

Autor: Christian Thiede, TACP Operations GmbH & Co. KG (Explicio), Detmold, ORCID 0009-0002-4517-7208. Bei Entwicklung, Messung, Auswertung und Text hat ein KI-Assistent (Claude, Anthropic) mitgearbeitet; die Verantwortung für Inhalt und Zahlen liegt beim Autor. Aussagen über Modelle Dritter (Qwen3.6, Nimble, Clef Flash) beziehen sich auf die genannten Fassungen, unsere Testsätze und das Datum der Messung. Das Dokument darf unverändert weitergegeben werden.



Explicio®

Das digitale Gedächtnis Ihres Unternehmens

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