f2b2

BAND 2

REDEN UND RECHNEN

Deine KI im Tagesgeschäft.
Dokumente befragen, Daten auswerten.
Die Daten bleiben im Haus.

Frank Tentler.ai · Stand 17.08.2026

f2b2

Eine Reihe für Menschen,
die keine Zeit für Vorworte haben.

WERKSTATTFASSUNG

Dieser Band trägt einen Stempel, den Band 1 nicht brauchte. Band 1 beschrieb eine Installation, und Installationen kann man vollständig recherchieren. Band 2 beschreibt Arbeitsweisen, und Arbeitsweisen kann man erst nach Wochen echter Nutzung endgültig beurteilen. Alles Technische hier ist am Erscheinungstag gegen die offizielle Dokumentation geprüft. Alles über Routinen ist begründete Empfehlung, die nach vier Wochen Praxis revidiert wird. Der Stempel verschwindet mit der zweiten Fassung.

f2b2 · Band 2
Reden und Rechnen
Stand: 17. August 2026
Setzt voraus: Band 1 (Modell läuft)
Frank Tentler.ai
[Created with human/ai co-intelligence]

INHALT

0

WAS DU HIER BRAUCHST

Eine Seite Voraussetzungen, dann geht es los

Band 1 ist durch, dein Modell läuft. Für diesen Band kommt eine Schicht dazu: Ollama als Motor im Hintergrund und Open WebUI als Arbeitsoberfläche davor. LM Studio bleibt installiert und ist weiter dein Werkzeug für schnelle Einzelfragen, aber die beiden Betriebsarten, um die es hier geht, wohnen in Open WebUI, weil dort Wissenssammlungen und Code-Ausführung eingebaut sind.

Die Begriffe für alles Weitere, jeweils in einem Satz. Reden heißt: Du legst Dokumente in Sammlungen ab und diskutierst mit dem Modell über deren Inhalt, mit Fundstellenangabe. Rechnen heißt: Das Modell schreibt auf deine Frage hin ein kleines Auswertungsprogramm, dein Mac führt es aus, und die Zahlen im Ergebnis stammen aus dem Programm, nicht aus dem Sprachgefühl des Modells. Warum diese Trennung über dein Vertrauen in jede einzelne Zahl entscheidet, ist der rote Faden dieses Bandes.

MACH ES GENAU SO

Installiere Ollama von `ollama.com` und lade das Modell wie in Band 1, Kapitel 6 beschrieben. Open WebUI installierst du am einfachsten über Docker Desktop; die jeweils aktuelle Befehlszeile dafür steht auf `docs.openwebui.com`, und sie ändert sich oft genug, dass sie nicht hier stehen sollte. Danach öffnest du im Browser `localhost:3000`, legst ein lokales Konto an und wählst dein Modell aus. Ab hier spielt alles Weitere in diesem Fenster.

HIER ENDET DIE FREUDE

Docker ist die erste echte Hürde dieses Bandes. Es ist ein Programm, das Programme in Containern verpackt, und es will nach der Installation einmal gestartet sein, bevor irgendein Befehl funktioniert. Die Fehlermeldung `Cannot connect to the Docker daemon` heißt übersetzt: Docker Desktop läuft gerade nicht. Starten, dreißig Sekunden warten, Befehl wiederholen. Diese eine Meldung produziert mehr Supportanfragen als der ganze Rest der Installation.

1

DEIN MODELL LERNT NICHTS. GUT SO.

Das Wort "trainieren" und was du stattdessen wirklich tust

Die verbreitetste Erwartung an eine eigene KI lautet: Sie lernt mich mit der Zeit kennen. Sie tut es nicht. Die Gewichte deines Modells sind seit dem Download eingefroren, jedes Gespräch beginnt bei null, und nichts, was du ihm erzählst, verändert es dauerhaft. Was nach Enttäuschung klingt, ist für deine Arbeit ein Vorteil: Ein Modell, das sich nicht verändert, kann sich auch nicht schleichend verschlechtern, nichts Falsches einprägen und keine Datenspuren in seinen Gewichten ansammeln. Es ist jeden Morgen exakt das geprüfte Werkzeug von gestern.

Echtes Nachtraining, das Fachwort ist Fine-Tuning, existiert, verlangt aber aufbereitete Datensätze, Rechenzeit und Erfahrung, und es lohnt sich für eine Einzelperson oder eine kleine Geschäftsstelle praktisch nie. Merk dir stattdessen den Satz, der dieses ganze Kapitel ersetzt: Du trainierst das Modell nicht, du briefst es. Und zwar bei jedem Start neu, über drei Kanäle, die zufällig die nächsten drei Kapitel dieses Bandes sind: eine dauerhafte Systemanweisung, die Rolle und Regeln festlegt. Deine eigenen Dokumente, in denen es nachschlagen kann. Und gute Beispiele in der Aufgabenstellung, an denen es sich orientiert.

Der Unterschied ist mehr als Wortklauberei. Wer glaubt, das Modell lerne, wird nachlässig beim Briefen und wundert sich über schwankende Ergebnisse. Wer weiß, dass jede Sitzung bei null beginnt, baut sein Wissen in wiederverwendbare Bausteine, und genau diese Bausteine sind es, die mit der Zeit besser werden. Nicht das Modell entwickelt sich, dein Werkzeugkasten tut es.

MACH ES GENAU SO

Leg dir vom ersten Tag an eine schlichte Textdatei namens `bausteine.txt` an, in der du drei Dinge sammelst: Formulierungen, die funktioniert haben. Aufgabenstellungen, die gute Ergebnisse brachten. Und Korrekturen, die du öfter als zweimal tippen musstest. Diese Datei ist dein eigentliches Trainingsergebnis, und sie gehört dir, nicht dem Modell.

HIER ENDET DIE FREUDE

Die Enttäuschung schlägt meist in Woche zwei zu, wenn das Modell zum dritten Mal nach der Vereinsstruktur fragt, die du ihm doch längst erklärt hast. Das ist kein Defekt und kein Grund für ein anderes Modell, das ist die Architektur. Die Antwort darauf steht im nächsten Kapitel und heißt Systemanweisung. Wer stattdessen anfängt, jedem Chat dieselben drei Absätze Vorgeschichte voranzustellen, macht nichts falsch, aber er macht Handarbeit, die das System ihm abnehmen kann.

2

DIE SYSTEMANWEISUNG

Das wichtigste Werkzeug des ganzen Bandes, und es ist reiner Text

Die Systemanweisung ist ein Text, den das Modell vor jedem Gespräch unsichtbar zu lesen bekommt. Sie legt fest, wer es sein soll, was es über deinen Betrieb wissen muss und nach welchen Regeln es arbeitet. In Open WebUI hinterlegst du sie einmal in den Einstellungen, wahlweise global oder als eigenes Arbeitsmodell mit eigenem Namen, und ab dann beginnt kein Gespräch mehr bei null. Das ist der Hebel mit dem besten Verhältnis aus Aufwand und Wirkung in diesem ganzen Buch: zwanzig Minuten Schreibarbeit, dauerhafte Wirkung.

Der Unterschied zwischen einer guten und einer wohlklingenden Systemanweisung entscheidet alles. Wohlklingend ist: "Du bist ein hilfreicher, professioneller Assistent für Stadtmarketing." Das ist Parfüm, es enthält keine einzige Information, die das Modell nicht schon hätte. Gut ist, was konkret und prüfbar ist: welche Organisation, welche Datenbestände mit welchen Eigenheiten, welche Ausgaberegeln, welche Verbote. Eine brauchbare Anweisung liest sich unspektakulär wie eine Arbeitsplatzbeschreibung, und genau so soll sie sich lesen.

EIN GERÜST ZUM AUSFÜLLEN

```
Du arbeitest für [Organisation, ein Satz].
Deine Hauptaufgaben: [zwei, drei konkrete Tätigkeiten].
Unsere Daten: [welche Tabellen und Berichte, welche
Eigenheiten, z.B. "Städtenamen sind uneinheitlich
geschrieben"].
Regeln: Antworte auf Deutsch. Wenn Information fehlt,
sag es, statt zu raten. Zahlen nur aus Berechnungen
oder Fundstellen, nie geschätzt. Bei Auswertungen:
nenne immer die Datenquelle und den Stand.
```

MACH ES GENAU SO

Schreib die erste Fassung in zwanzig Minuten und betrachte sie als Verschleißteil. Immer wenn du dich dabei ertappst, dem Modell zum wiederholten Mal dasselbe zu erklären, wandert genau dieser Satz aus dem Chat in die Systemanweisung. Nach vier Wochen hat sie sich auf diese Weise selbst geschrieben, und zwar aus deiner echten Arbeit statt aus einer Vorlage.

HIER ENDET DIE FREUDE

Die überladene Anweisung ist genauso schädlich wie die leere. Wer dreißig Regeln hineinschreibt, stellt fest, dass das Modell die wichtigen fünf schlechter befolgt, weil sie zwischen fünfundzwanzig unwichtigen stehen. Ein 27B-Modell ist kein Jurist, der Paragrafen abarbeitet. Faustregel für diese Modellklasse: eine halbe Seite, die zehn wichtigsten Sätze, und jede Regel, die in vier Wochen nie relevant wurde, fliegt wieder raus.

3

DAS BOOTDOC

Dein Briefing als Datei, die mit dir umzieht

Die Systemanweisung aus dem letzten Kapitel hat eine stille Schwäche: Sie wohnt im Werkzeug. Wechselst du von Open WebUI zu LM Studio, sitzt du am Rechner einer Kollegin oder probierst nächstes Jahr ein neues Programm aus, beginnt alles wieder bei null, und dein sorgfältig gebautes Briefing bleibt im alten Menü zurück. Die Lösung ist so naheliegend, dass man sie leicht übersieht: Das Briefing wird selbst ein Dokument. Wir nennen es Bootdoc, weil es die KI startklar macht wie ein Startvorgang den Rechner.

Ein Bootdoc ist eine gewöhnliche Text- oder PDF-Datei mit zwei Schichten, die offen nebeneinander liegen. Die eine Schicht ist für Menschen geschrieben und erklärt in Ruhe, wie eine bestimmte Arbeit in deinem Haus funktioniert, etwa dein Newsletter, deine Monatsauswertung, dein Berichtswesen. Die andere Schicht besteht aus kurzen, sichtbaren Konfigurationsblöcken, in denen steht, was die Maschine mit alldem anfangen soll: welche Regeln gelten, welche Begriffe wie geschrieben werden, was sie nie behaupten darf. Nichts davon ist versteckt. Wer das Dokument liest, sieht exakt, was die KI erfährt, und diese Sichtbarkeit ist kein Nebeneffekt, sondern der Punkt: Ein Briefing, das man vorzeigen kann, ist ein Briefing, dem man trauen kann.

Benutzt wird es mit einer Aktivierungsformel, einem festen Satz am Gesprächsanfang: Datei in den Chat, dazu die Anweisung, das Dokument vollständig zu lesen und ab jetzt auf Basis seiner Inhalte und Regeln zu arbeiten. Ab diesem Moment kennt jedes System deinen Arbeitskontext, egal welches, und der Nullstart aus Kapitel 1 ist Geschichte. Der zweite Gewinn zeigt sich bei der Weitergabe: Übernimmt jemand deine Aufgabe, im Urlaub oder auf Dauer, bekommt er kein Übergabegespräch mit zwanzig Punkten, sondern eine Datei, die antworten kann. Was danach noch unklar ist, sind erfahrungsgemäß zwei Fragen statt zwanzig.

MACH ES GENAU SO

Bau dein erstes Bootdoc für genau einen Arbeitsbereich, nicht für alles auf einmal, nach diesem Gerüst: ein erzählender Teil, der die Arbeit erklärt, als würdest du sie einer neuen Kollegin zeigen. Dann ein Konfigurationsblock mit den Maschinenregeln: Kontext in einem Satz, Schreibweisen, Ausgaberegeln, Verbote, jeweils eine Zeile. Zum Schluss die Aktivierungsformel zum Kopieren: "Lies dieses Dokument vollständig. Arbeite ab jetzt auf Basis seiner Inhalte und Konfigurationsblöcke. Halte dich an die Regeln, auch wenn ich nicht daran erinnere." Für lokale Modelle gilt das Schlankheitsgebot aus Kapitel 2 doppelt: alle Konfigurationsblöcke zusammen höchstens eine halbe Seite, der erzählende Teil darf länger sein, denn er dient dir und der Betriebsart Reden.

HIER ENDET DIE FREUDE

Zwei Arten, wie dir das Bootdoc um die Ohren fliegt. Die erste ist Übergewicht: Ein 27B-Modell folgt einem Regelwerk von acht Seiten nicht zuverlässig, es vergisst still einzelne Regeln, und du merkst es erst am Ergebnis. Wenn Regeln nicht greifen, kürze das Bootdoc, bevor du am Modell zweifelst. Die zweite ist gefährlicher: Ein Bootdoc konfiguriert deine KI, also kann ein fremdes sie auch fehlkonfigurieren, mit Regeln, die du beim Überfliegen nicht bemerkst. Behandle fremde Bootdocs deshalb wie fremde Dateianhänge. Vor dem Laden gilt die Dreifachprüfung: bekannte Quelle, nachvollziehbarer Inhalt, sichtbare Konfiguration. Ein Bootdoc mit verstecktem oder unklarem Regelteil lädst du nicht, egal wie nützlich es klingt.

4

BETRIEBSART REDEN: DOKUMENTE BEFRAGEN

Wissenssammlungen, Fundstellen und die Grenze des Verfahrens

In Open WebUI heißt der Ort dafür Knowledge, zu Deutsch Wissen. Du legst dort Sammlungen an, etwa "Jahresberichte", "Protokolle", "Positionspapiere", und füllst sie mit Dateien. Stellst du im Chat eine Frage und bindest eine Sammlung ein, sucht das System die inhaltlich passenden Textstellen heraus, reicht sie dem Modell und das Modell antwortet mit Bezug auf die Fundstelle. Das Fachwort für dieses Nachschlagen heißt RAG, und der einzige Satz, den du dir dazu merken musst: Das Modell liest nicht deine ganze Sammlung, es liest die Stellen, die die Suche ihm vorlegt.

Aus diesem Satz folgen die beiden Betriebsregeln. Erstens funktioniert das Verfahren umso besser, je klarer deine Frage von Begriffen handelt, die auch im Dokument stehen; wer nach "dem Ding mit den Innenstädten von neulich" fragt, gibt der Suche nichts zu greifen. Zweitens bietet Open WebUI pro Dokument zwei Modi an: die fokussierte Suche für große Sammlungen und den Vollkontext-Modus, der ein einzelnes Dokument komplett ins Gespräch holt. Für die Diskussion über einen einzelnen Bericht ist der Vollkontext die verlässlichere Wahl, weil nichts durch das Suchraster fallen kann; dein Kontextfenster aus Band 1 muss dann groß genug eingestellt sein.

MACH ES GENAU SO

Bau die erste Sammlung klein: fünf bis zehn Dokumente, ein Thema, sprechende Dateinamen wie `jahresbericht-2025.pdf` statt `final_v3_neu2.pdf`, denn die Namen tauchen in den Fundstellen wieder auf. Stell dann fünf Fragen, deren Antworten du selbst kennst, und prüfe die Fundstellen. Erst wenn diese Stichprobe sitzt, wächst die Sammlung. Eine Sammlung, der du nicht traust, ist wertlos, egal wie groß sie ist.

HIER ENDET DIE FREUDE

Die Betriebsart Reden hat eine eingebaute Grenze, und an ihr entsteht der gefährlichste Fehler dieses Bandes: Sie kann Textstellen finden, aber nicht rechnen. Fragst du eine Sammlung voller Statistikberichte "wie haben sich die Frequenzen im Schnitt entwickelt", bekommst du eine flüssig formulierte Zahl, die aus gefundenen Textfetzen zusammengereimt ist. Sie kann stimmen. Sie kann auch souverän klingender Unsinn sein, und du siehst ihr den Unterschied nicht an. Jede Frage, in der gezählt, gemittelt, verglichen oder verschnitten wird, gehört in die andere Betriebsart, und die kommt jetzt.

5

BETRIEBSART RECHNEN: ZAHLEN AUS SKRIPTEN

Warum eine KI, die rechnet, verdächtig ist, und eine, die rechnen lässt, brauchbar

Ein Sprachmodell erzeugt Text, der plausibel klingt, und genau deshalb darfst du ihm beim Kopfrechnen nicht trauen: Eine erfundene Summe klingt exakt so überzeugend wie eine richtige. Die Lösung ist eine saubere Arbeitsteilung. Das Modell schreibt auf deine Frage hin ein kleines Python-Programm, dein Mac führt es aus, und das Programm liefert das Ergebnis. Programme verrechnen sich nicht, sie sind wiederholbar, und man kann sie lesen und prüfen. Das Modell wechselt vom Rechenknecht, der es nicht sein kann, zum Programmierer und Erklärer, der es sein kann.

In Open WebUI heißt diese Fähigkeit Code Interpreter. Du aktivierst sie beim Chat, lädst deine Tabelle hoch und fragst in normalem Deutsch: "Lies die Datei ein, verbinde sie über die Spalte Stadt mit der zweiten Tabelle und zeig mir die zehn größten Abweichungen." Das Modell schreibt das Skript, die Ausführung läuft lokal, und du bekommst Tabelle, Grafik und den Programmcode dazu. Den Code musst du nicht verstehen, aber wirf einen Blick darauf, ob die richtigen Spalten vorkommen; das ist wie der Blick auf den Kassenbon, man muss nicht Buchhalter sein, um "Spalte Umsatz fehlt" zu bemerken.

Zur Einordnung der Kraft deines Modells: Standardauswertungen, also einlesen, filtern, gruppieren, verbinden, zusammenfassen, darstellen, beherrscht die 27B-Klasse ordentlich. Bei verschachtelten Sonderwünschen wird sie Fehler produzieren, und was dann hilft, steht im nächsten Kapitel.

MACH ES GENAU SO

Starte mit dem eingebauten Ausführungsmodus, der ohne weitere Installation läuft, und einer kleinen Testtabelle, deren Ergebnisse du kennst. Für ernsthafte Auswertungen mit großen Dateien empfiehlt die Open-WebUI-Dokumentation inzwischen das Terminal-Backend im Docker-Container, das jede Python-Bibliothek nutzen kann; das ist ein Ausbau-Schritt für später, nicht für den ersten Tag. Und gewöhn dir von Anfang an die Kontrollfrage an: "Woher stammt diese Zahl?" Die richtige Antwort zeigt auf eine Skriptausgabe, nie auf das Modell selbst.

HIER ENDET DIE FREUDE

Der Bruch entsteht, wenn das Modell heimlich zurück in die Plauderrolle fällt. Es passiert bei Anschlussfragen: Die erste Auswertung lief sauber übers Skript, dann fragst du "und wie sieht das für die kleineren Städte aus", und das Modell antwortet aus dem Gedächtnis des Gesprächs, ohne neu zu rechnen. Erkennungszeichen: Es erscheint kein neuer Codeblock. Die Gegenmaßnahme ist ein Satz in der Systemanweisung aus Kapitel 2: "Jede Zahl in deinen Antworten stammt aus einer ausgeführten Berechnung; wenn keine Berechnung lief, rechne neu statt zu schätzen."

6

DIE ZWEI-ZONEN-REGEL

Eine Grenze, die jeder versteht: Daten bleiben im Haus, Strukturen dürfen reisen

Vermutlich nutzt du neben deiner lokalen Installation weiter einen Cloud-Dienst, und das ist kein Stilbruch, sondern vernünftig. Damit daraus kein Datenschutzunfall wird, brauchst du keine dreißigseitige Richtlinie, sondern eine Grenze, die sich in einem Satz sagen lässt: Alles, was Inhalte enthält, berührt nur den eigenen Rechner. Die Cloud sieht höchstens Strukturen.

Was der Unterschied ist, zeigt eine Tabelle. Ihre Inhalte, also Zeilen mit Städtenamen, Zahlen, Personen, sind Daten und bleiben im Haus. Ihre Struktur, also die Information "Spalte 1 heißt Stadt, Spalte 2 heißt Passantenfrequenz, Spalte 3 heißt Monat", ist kein schützenswertes Geheimnis. Und genau diese Unterscheidung macht die Cloud als Werkstatt nutzbar, ohne sie als Datenspeicher zu missbrauchen: Wenn dein lokales Modell an einer komplizierten Auswertung scheitert, beschreibst du einem Cloud-Modell nur die Spaltenstruktur und die gewünschte Auswertung, lässt dir das Skript schreiben und führst es zu Hause auf den echten Daten aus. Das große Modell liefert die Programmierkunst, deine Daten sieht es nie, und das fertige Skript arbeitet ab dann dauerhaft lokal.

MACH ES GENAU SO

Formuliere die Anfrage an die Cloud nach diesem Muster: "Schreib mir ein Python-Skript mit Pandas. Tabelle A hat die Spalten [Namen], Tabelle B hat die Spalten [Namen]. Ich will [Auswertung]. Die Daten liegen als Excel-Dateien vor." Kopiere das Skript nach Hause, lass es dein lokales System ausführen und leg es in einen Ordner `skripte`, den Kapitel 8 gleich zum Werkzeugkasten befördert.

HIER ENDET DIE FREUDE

Die Regel stirbt nicht am bösen Willen, sondern an der Bequemlichkeit im Einzelfall: "Nur diese eine Tabelle schnell hochladen, ist ja fast nichts drin." Beim dritten Mal ist die Grenze Gewohnheit geworden, nur eben die falsche. Deshalb gehört zur Zwei-Zonen-Regel eine Krücke für den Moment der Versuchung: Wenn es wirklich schnell gehen muss, lösche vor dem Hochladen alle Datenzeilen bis auf zwei und ersetze deren Inhalte durch erfundene. Struktur mit Beispielcharakter darf reisen, und die Versuchung hat einen legalen Ausweg.

7

PDFS, DIE HÄSSLICHSTE ECKE

Warum die Tabelle im Bericht keine Tabelle ist

Ein PDF ist kein Datenformat, es ist gedrucktes Papier, das zufällig auf einem Bildschirm liegt. Die Tabelle, die du darin siehst, existiert für den Computer nicht als Tabelle, sondern als lose Wolke aus Textschnipseln mit Positionsangaben. Für die Betriebsart Reden ist das halb so schlimm, Fließtext aus PDFs funktioniert ordentlich. Für die Betriebsart Rechnen ist es das größte einzelne Ärgernis dieses Bandes: Bevor irgendetwas ausgewertet werden kann, muss aus der Textwolke wieder eine echte Tabelle werden, und dieser Schritt geht schief, ohne sich zu melden. Verrutschte Spalten, verschluckte Kopfzeilen, aus 1.234 wird 1234 oder schlimmer 1,234.

Die Konsequenz ist eine Rangfolge, die du dir angewöhnen solltest, bevor der Frust sie dir beibringt. Erste Wahl: die Originaldatei. Fast jedes Tabellen-PDF war einmal eine Excel-Datei, und eine Mail an den Absender mit der Bitte um das Original spart Stunden. Zweite Wahl: die Extraktion als eigener, kontrollierter Arbeitsschritt, bei dem das Skript zunächst nur die Tabelle herauslöst und als Excel- oder CSV-Datei speichert, die du einmal per Auge prüfst. Erst danach beginnt die eigentliche Auswertung, und zwar auf der geprüften Datei. Dritte Wahl, für Gescannte und Fotografierte: gar nicht erst automatisieren, sondern die paar Werte abtippen oder das Dokument als Fall für die Betriebsart Reden akzeptieren.

MACH ES GENAU SO

Formuliere den Auftrag an dein Modell immer zweistufig: "Extrahiere zuerst nur die Tabelle ab Seite 12 und speichere sie als `tabelle_roh.xlsx`. Werte noch nichts aus." Dann öffnest du die Datei, vergleichst zwanzig Sekunden lang Stichproben mit dem PDF und gibst erst dann den Auswertungsauftrag. Diese zwanzig Sekunden sind die beste Zeitinvestition des ganzen Verfahrens.

HIER ENDET DIE FREUDE

Der stille Extraktionsfehler ist deshalb so gefährlich, weil alles danach fehlerfrei aussieht. Das Skript läuft, die Grafik ist hübsch, die Zahlen sind präzise, nur eben präzise falsch, weil in Zeile 40 des PDFs eine Spalte verrutscht ist. Gewöhn dir eine Plausibilitätsfrage pro Auswertung an: eine Zahl, deren Wert du unabhängig kennst, etwa die Einwohnerzahl deiner eigenen Stadt. Stimmt sie nicht, ist die Extraktion schuld, nicht die Statistik.

8

AUS FRAGEN WERDEN KNÖPFE

Der Moment, in dem aus Spielerei ein Arbeitsgerät wird

Der Unterschied zwischen Ausprobieren und Arbeiten ist Wiederholbarkeit. Die Monatsauswertung, die du im März mühsam erfragt hast, darf im April keine neue Verhandlung mit dem Modell sein, sie ist ein gespeichertes Skript, das mit der neuen Datei einfach noch einmal läuft. Gleiche Eingabe, gleiches Verfahren, vergleichbares Ergebnis, und genau die Vergleichbarkeit über Monate ist bei wiederkehrenden Auswertungen der halbe Wert.

Deshalb ist der Ordner `skripte` aus Kapitel 6 kein Ablagekasten, sondern dein wachsendes Arbeitsgerät. Jedes Skript, das sich einmal bewährt hat, bekommt einen sprechenden Namen wie `monatsvergleich_frequenzen.py` und zwei Kommentarzeilen am Anfang: was es tut und welche Eingabedateien es erwartet. Für die Wiederholung brauchst du dann kein neues Modell-Gespräch, sondern sagst nur: "Führe monatsvergleich_frequenzen.py mit der Datei april.xlsx aus." Dasselbe Prinzip gilt für die Betriebsart Reden: Wiederkehrende Aufgabenstellungen, etwa dein Muster für Protokollzusammenfassungen, wandern als Textbausteine in die Sammlung aus Kapitel 1 oder gleich als Prompt-Vorlage in Open WebUI.

MACH ES GENAU SO

Die Beförderungsregel: Was du zum zweiten Mal brauchst, wird beim zweiten Mal zum Werkzeug gemacht, nicht erst beim fünften. Beim zweiten Mal weißt du noch, was beim ersten Mal hakte, und die fünf Minuten fürs Benennen und Kommentieren sind billiger als jede spätere Rekonstruktion. Einmal im Monat wirfst du einen Blick in den Ordner und löschst, was sich nicht bewährt hat; ein Werkzeugkasten mit vierzig Werkzeugen, von denen du sechs benutzt, ist ein Suchproblem.

HIER ENDET DIE FREUDE

Die Falle heißt Drift: Die Aprildatei hat plötzlich eine Spalte mehr als die Märzdatei, das gespeicherte Skript läuft trotzdem durch und rechnet still das Falsche. Robuste Skripte prüfen deshalb am Anfang, ob die erwarteten Spalten da sind, und brechen sonst mit einer klaren Meldung ab. Diesen Prüfblock schreibt dir dein Modell auf Zuruf dazu: "Ergänze am Anfang eine Prüfung, ob die Spalten Stadt, Monat und Frequenz existieren, und brich sonst mit einer verständlichen Fehlermeldung ab." Ein Skript, das laut scheitert, ist Gold. Eines, das leise falsch rechnet, ist der Grund, warum Verbände Excel-Ergebnisse anzweifeln.

9

DIE STRICHLISTE

Vier Wochen Messung statt Meinung, und der Mut zum Seinlassen

Nach der Einrichtung beginnt die Phase, für die es keine Anleitung geben kann, weil sie von deiner Arbeit handelt. Was es geben kann, ist ein Messverfahren. Führe vier Wochen lang eine Strichliste mit drei Spalten: Aufgaben, die lokal gut liefen. Aufgaben, bei denen du zur Cloud gewechselt hast, mit einem Wort zum Grund. Und Aufgaben, die du ganz aufgegeben hast. Nach zwanzig Einträgen hast du etwas, das kein Benchmark der Welt dir geben kann: die Arbeitsteilung, die zu deinen Texten, deinen Tabellen und deiner Geduld passt.

Genauso wichtig wie das Dazulernen ist das Seinlassen, und dieser Absatz fehlt in jedem Ratgeber dieser Sorte. Es ist kein Scheitern, wenn nach vier Wochen feststeht, dass deutsche Publikationsprosa in der Cloud bleibt, dass das Diskutieren über Dokumente seltener vorkommt als gedacht oder dass eine der eingerichteten Fähigkeiten schlicht ungenutzt blieb. Jede bewusst gestrichene Funktion macht das System einfacher, und einfachere Systeme werden benutzt. Das Gegenteil, ein voll ausgebautes System, das niemand anfasst, ist das häufigste Ende solcher Projekte, und es beginnt fast immer damit, dass niemand den Mut zum Streichen hatte.

Die Strichliste hat noch einen zweiten Adressaten: die nächste Fassung dieses Bandes. Der Werkstattstempel auf Seite 2 verschwindet, wenn aus vier Wochen Praxis feststeht, welche Kapitel sich bewährt haben und welche Empfehlung revidiert gehört.

MACH ES GENAU SO

Nimm für die Liste das dümmste Werkzeug, das du hast: ein Blatt neben der Tastatur oder eine Notiz auf dem Telefon. Jede Lösung, die selbst Einrichtung braucht, wird nicht geführt. Trag nichts nach; was du abends nicht mehr weißt, war keinen Strich wert.

HIER ENDET DIE FREUDE

Der Selbstbetrug in dieser Phase hat ein Muster: Man gibt dem lokalen System die Aufgaben, von denen man weiß, dass es sie schafft, und feiert die Quote. Die Liste wird erst ehrlich, wenn auch die Wechsel zur Cloud und die Abbrüche drinstehen, und zwar am selben Tag. Eine geschönte Strichliste ist keine Messung, sie ist Marketing an dich selbst, und du bist der einzige Kunde, der darauf hereinfällt.

10

WAS ZUERST VERFÄLLT

Die Haltbarkeitstabelle dieses Bandes

Zuerst verfallen die Oberflächendetails von Open WebUI: wo welcher Schalter sitzt, wie die Ausführungsmodi heißen, welcher Installationsbefehl gilt. Das Projekt entwickelt sich schnell, und genau deshalb steht in diesem Band kein einziger Klickpfad mit Screenshot, sondern immer der Name der Funktion, nach dem du in der jeweils aktuellen Dokumentation suchen kannst. Knowledge, Code Interpreter, Systemanweisung, diese Begriffe überleben ihre Menüposition.

Danach verfällt die Einschätzung der Modellklasse. Dass ein 27B-Modell Standardauswertungen schafft und bei komplexen Verschneidungen Hilfe aus der Cloud braucht, ist eine Momentaufnahme vom August 2026. Die Grenze wandert mit jeder Modellgeneration nach oben, die Zwei-Zonen-Regel, das Bootdoc und der Schema-Trick funktionieren aber auf jeder Höhe dieser Grenze weiter, sie werden nur seltener nötig.

Am längsten halten die drei Sätze, auf denen der ganze Band steht. Du trainierst das Modell nicht, du briefst es. Zahlen kommen aus Skripten, nicht aus Sprachgefühl. Und Daten bleiben im Haus, während Strukturen reisen dürfen. Wer nur diese drei Sätze behält und den Rest vergisst, arbeitet immer noch richtig, nur langsamer.

MACH ES GENAU SO

Bei der Weitergabe dieses Bandes gilt derselbe Satz wie bei Band 1, nur mit anderen Kapitelnummern: "Stand August 2026, die Kapitel 1, 6 und 9 stimmen noch, die Werkzeugdetails musst du prüfen." Und der Werkstattstempel gehört dazuerzählt: Dieser Band ist vor der Praxis geschrieben und wird an ihr korrigiert.

Damit hast du beide Betriebsarten, die Grenze dazwischen und ein Messverfahren für alles Weitere. Der Rest entsteht nicht beim Lesen, sondern neben der Tastatur, auf einem Blatt mit Strichen darauf.

Frank Tentler.ai

[Created with human/ai co-intelligence]

f2b2

REDEN MIT DOKUMENTEN. RECHNEN MIT SKRIPTEN. DIE DATEN BLEIBEN IM HAUS.

Band 1 hat das Modell auf deinen Mac gebracht. Band 2 spannt es ein: Dokumente befragen mit Fundstellen, Tabellen und Statistiken auswerten mit Zahlen, denen du trauen kannst, wiederkehrende Analysen als gespeicherte Werkzeuge. Dazu die eine Regel, die aus dem Ganzen eine saubere Sache macht: Daten bleiben im Haus, Strukturen dürfen reisen.

Eine Reihe für Menschen, die keine Zeit für Vorworte haben.

f2b2 · Band 2 · Werkstattfassung · Stand 17.08.2026
Frank Tentler.ai · [Created with human/ai co-intelligence]