Vom Notebook zur Anwendung: Was der Aufbau des M-Modells über KI-Projekte lehrt

Wer heute eine KI-gestützte Anwendung bauen will, landet meist schnell bei derselben Frage: Fängt man mit der fertigen Architektur an – oder mit dem einzelnen Rechenschritt, den man verstehen will? Am Beispiel des M-Modells, meiner Pipeline zur automatisierten Bewertung von Start-ups und KMU, will ich zeigen, warum die zweite Antwort die belastbarere ist – und was das für die Frage bedeutet, wie viel „Intelligenz" ein solches System eigentlich braucht.

Die Ausgangsidee

Das M-Modell nimmt zwei Dokumente entgegen – einen Businessplan und eine strukturierte Finanzplanung – und errechnet daraus einen Investment Score zwischen 0 und 100, dazu einen ausformulierten Bericht. Das System läuft vollständig lokal, ohne Cloud-Anbindung; bei vertraulichen Unternehmensdaten ist das keine Nebensache, sondern Voraussetzung.

Interessanter als das Ergebnis ist für mich als Forscher der Weg dorthin gewesen. Die erste Fassung bestand nicht aus einer Anwendung, sondern aus sieben aufeinanderfolgenden Jupyter Notebooks – jedes für genau einen Analyseschritt: Text extrahieren, qualitativ bewerten, Kennzahlen berechnen, Plausibilität prüfen, Score bilden, Szenarien simulieren. Jedes Notebook schrieb sein Ergebnis als Datei auf die Festplatte, bevor das nächste sie einlas.

Der eigentliche Befund: Die Struktur entsteht im Notebook, nicht in der App

Das lässt sich leicht als Übergangsphase abtun – „man hat halt erst mal experimentiert, bevor man es sauber gebaut hat". Der eigentliche Befund ist aber ein anderer: Als die Logik später in eine durchgehende Web-Anwendung überführt wurde, wurde die Modulstruktur nicht neu entworfen. Aus jedem Notebook wurde nahezu unverändert ein eigenständiges Python-Modul, und die Dateischnittstelle, über die sich die Notebooks zuvor Ergebnisse zugereicht hatten, blieb exakt die Schnittstelle, über die die Module bis heute miteinander sprechen.

Der Wechsel betraf also nicht die Analyselogik, sondern nur ihre Ansteuerung: Aus sieben manuell zu startenden Notebooks wurde ein einziger Knopfdruck. Und genau diese Kontinuität ist der Grund, warum sich die Anwendung später so unkompliziert erweitern ließ – um eine automatische Dokumentensuche, einen KI-Textbaustein, eine Prüfroutine für die Eingabedaten, einen automatischen Berichtsexport. Jede Erweiterung ließ sich als neues, in sich abgeschlossenes Modul einfügen, ohne die bestehenden anzufassen, weil die Schnittstellen von Anfang an sauber geschnitten waren.

Für mich ist das die eigentliche methodische Erkenntnis dieses Projekts: Man muss die spätere Softwarearchitektur nicht vorab entwerfen. Es genügt, jeden Denkschritt zunächst als eigenständigen, klar abgegrenzten Notebook-Abschnitt mit einer sauberen Ergebnisdatei zu erproben. Was sich dort stabilisiert, wird fast unverändert zum Baustein der Anwendung.

Standardisierung vor Intelligenz – diesmal mit Hardware-Beleg

Das Leitprinzip, das ich in früheren Beiträgen bereits skizziert habe, bekommt in diesem Projekt eine sehr konkrete Begründung: Die verwendete Maschine hat rund 15,4 GB

Arbeitsspeicher und keine eigenständige Grafikkarte. Alle Sprachmodelle laufen also über den normalen Prozessor – spürbar langsamer, und praktisch beschränkt auf Modelle der 7–12-Milliarden-Parameter-Klasse.

Diese Grenze ist keine Randnotiz, sondern die eigentliche Begründung der Architektur: Weil die Leistungsfähigkeit des lokalen Sprachmodells begrenzt ist, muss der rein rechnerische Kern – Kennzahlen, Score, Plausibilitätsprüfung – die Hauptarbeit tragen. Das Sprachmodell liefert Sprache und Einordnung, nicht die Entscheidungsgrundlage. Wer nur auf ein leistungsfähigeres Modell wartet, statt die Dateneingabe zu standardisieren, optimiert am falschen Ende.

Der Funktionsnachweis

Drei vollständig durchlaufene Testfälle aus unterschiedlichen Branchen – Logistik (71/100), Telematik (69/100), Gesundheitswesen (72/100) – bestätigen: Die Grundidee trägt auch dann noch zuverlässig, wenn ein System um KI-gestützte Bausteine erweitert wird, solange der zahlengetriebene Kern deterministisch bleibt.

Fazit

Das M-Modell ist weniger ein fertiges Produkt als ein laufendes Experiment darüber, wie viel Struktur man vorab braucht, um mit KI-Bausteinen sinnvoll zu arbeiten – und wie viel davon sich einfach aus sauberer, schrittweiser Erprobung ergibt. Die Notebook-Phase war dabei kein Umweg, sondern die eigentliche Entwurfsarbeit.